A type-based termination criterion for dependently-typed higher-order rewrite systems

Frédéric Blanqui

A type-based termination criterion for dependently-typed higher-order rewrite systems

Frédéric Blanqui *

January 22, 2004

Abstract: *Several authors devised type-based termination criteria for ML-like languages (polymorphic λ-calculi with inductive types and case analysis), that allows non-structural recursive calls. We extend these works to general rewriting and dependent types, hence providing a powerful termination criterion for the combination of rewriting and β-reduction in the Calculus of Constructions.*

1 Introduction

The Calculus of Constructions [19] is a powerful type system allowing polymorphic and dependent types. It is the basis of many proof assistants since it allows one to formalize the proofs of higher-order logic. In this context, it is essential to allow users to define functions and predicates in the most convenient way and to be able to decide whether a term is a proof of some proposition, and whether two terms/propositions are equivalent w.r.t. user definitions. As exemplified in [21, 11], a promising approach is rewriting. To this end, we need powerful criteria to check the termination of higher-order rewrite-based definitions combined with β-reduction.

In a previous work [11], we proved that such a combination is strongly normalizing if, on the one hand, first-order rewrite rules are strongly normalizing and non-duplicating[1] and, on the other hand, non first-order rewrite rules (called higher-order in the following) satisfies a termination criterion based on the notion of computability closure and similar to higher-order primitive recursion. Unfortunately, many interesting rewrite systems are either first-order and duplicating, or higher-order with non-structural recursive calls (*e.g.* division on natural numbers[2][3], Figure 1).

*Laboratoire Lorrain de Recherche en Informatique et Automatique (LORIA) & Institut National de Recherche en Informatique et Automatique (INRIA), 615 rue du Jardin Botanique, BP 101, 54602 Villers-lès-Nancy, France, `blanqui@loria.fr`.

[1] Strong normalization is not modular in general [38]. It is modular for non-duplicating first-order rewrite systems [35]. Here, we do not have two non-duplicating first-order rewrite systems but a hierarchical combination of a higher-order rewrite system (satisfying strong termination conditions) built over a non-duplicating first-order rewrite system.

[2] $/\ x\ y$ denotes $\lceil \frac{x}{y+1} \rceil$.

[3] We use curried symbols all over the paper.

Figure 1: Division on natural numbers

$$
\begin{array}{lrcl}
(1) & -\ x\ 0 & \to & x \\
(2) & -\ 0\ x & \to & 0 \\
(3) & -\ (sx)\ (sy) & \to & -\ x\ y \\
(4) & /\ 0\ x & \to & 0 \\
(5) & /\ (sx)\ y & \to & s\ (/\ (-\ x\ y)\ y)
\end{array}
$$

Hughes *et al* [28], Xi [41, 42], Giménez *et al* [26, 5] and Abel [2] devised termination criteria able to treat such examples by exploiting the way inductive types are usually interpreted [31]. Take for instance the addition[4] on Brouwer's ordinals *ord* (Figure 2) whose constructors are $0 : ord$, $s : ord \Rightarrow ord$ and $lim : (nat \Rightarrow ord) \Rightarrow ord$.

Figure 2: Addition on Brouwer's ordinals

$$
\begin{array}{lrcl}
(1) & +\ 0\ x & \to & x \\
(2) & +\ (sx)\ y & \to & s\ (+\ x\ y) \\
(3) & +\ (lim\ f)\ y & \to & lim\ ([x : nat](+\ (f\ x)\ y))
\end{array}
$$

The usual computability-based technique for proving the termination of this function is to interpret *ord* by the fixpoint of the following monotone function φ on the powerset of $\mathcal{SN}$, the set of strongly normalizing terms, ordered by inclusion:[5]

$$\varphi(X) = \{t \in \mathcal{SN} \mid t \to^* su \Rightarrow u \in X; t \to^* limf \Rightarrow \forall u \in \mathcal{SN}, fu \in X\}$$

The fixpoint of φ, $[\![ord]\!]$, can be reached by transfinite iteration and every $t \in [\![ord]\!]$ is obtained after a smallest ordinal $o(t)$ of iterations, the order of t. This naturally defines an ordering: $t > u$ iff $o(t) > o(u)$, with which we clearly have $lim\ f > fu$ for all $u \in \mathcal{SN}$.

Now, applying this technique to *nat*, we can easily check that $o(-tu) \leq o(t)$ and thus allow the recursive call with $-xy$ in the definition of /. First note that $-tu$ is computable (*i.e.* belongs to $[\![nat]\!]$) iff all its reducts are computable (see Section 5). We proceed by induction on $o(t)$:

- If $-tu$ matches rule (1) then $o(-tu) = o(t)$.
- If $-tu$ matches rule (2) then $o(-tu) = 0 \leq o(t)$.
- If $-tu$ matches rule (3) then $t = st'$ and $u = su'$. By induction hypothesis, $o(-t'u') \leq o(t')$. Thus, $o(-tu) = 1 + o(-t'u') \leq 1 + o(t') = o(t)$.
- If $-tu$ matches no rule then $o(-tu) = 0 \leq o(t)$.

[4] $[x : T]u$ denotes the function which associates u to every x of type T.

[5] $\to^*$ is the reflexive and transitive closure of the reduction relation $\to$.

The idea of the previously cited authors is to add this size/index/stage information to the syntax in order to prove this automatically. Instead of a single type nat, they consider a family of types $\{nat^{\mathfrak{a}}\}_{\mathfrak{a}\in\omega}$, each type $nat^{\mathfrak{a}}$ being interpreted by the set obtained after $\mathfrak{a}$ iterations of the function φ for nat. And they define a decidable type system in which minus (defined by *fixpoint/cases* constructions in their work) can be typed by $nat^{\alpha} \Rightarrow nat^{\beta} \Rightarrow nat^{\alpha}$, where α and β are size variables, meaning that the order of $-tu$ is not greater than the order of t.

This can also be interpreted as a way to automatically prove theorems on the size of the result of a function w.r.t. the size of its arguments [39, 25] with application to complexity and resource bound certification, and compilation optimization (*e.g.* bound check elimination [34], vector-based memoisation [16]).

In this paper, we extend this technique to the full Calculus of Algebraic Constructions [11] whose type conversion rule depends on the user-defined rewrite rules, and to general rewrite-based definitions (including matching on defined symbols and rewriting modulo equational theories [9]) instead of definitions only based on *letrec/match* (or *fixpoint/cases*) constructions. Note that our work makes a heavy use of (and simplify) the techniques developed by Chen for studying the Calculus of Constructions with subtyping [15].

On the one hand, we allow a richer size algebra than the one in [28, 5, 2] (see Section 6). On the other hand, we do not allow existential size variables and conditional rewriting[6] that are essential for capturing, for instance, the size-preserving property of quicksort (Example 5) and Mac Carty's "91" function (Example 8) respectively, as it can be done in Xi's work [42]. Note however that Xi is interested in the call-by-value normalization of closed simply-typed λ-terms, while we are interested in the strong normalization of the open terms of the Calculus of Constructions.

2 The Calculus of Algebraic Constructions with Size Annotations

The Calculus of Constructions (CC) is the full Pure Type System with the set of *sorts* $\mathcal{S} = \{\star, \Box\}$ and the axiom $\star : \Box$ [4]. $\star$ is intended to be the universe of types and propositions, while $\Box$ is intended to be the universe of predicate types. Let $\mathcal{X}$ be the set of variables.

The Calculus of Algebraic Constructions (CAC) [11] is an extension of CC with a set $\mathcal{F}$ of function or predicate *symbols* defined by a set $\mathcal{R}$ of (higher-order) rewrite rules [20, 30]. Every variable x (resp. symbol f) is equipped with a sort s_x (resp. s_f). We denote by $\mathcal{DF}$ the set of *defined* symbols, that is, the set of symbols f such that there is a rule $l \to r \in \mathcal{R}$ with $l = f\vec{l}$, and by $\mathcal{CF}$ the set $\mathcal{F} \setminus \mathcal{DF}$ of *constant* symbols. We add a superscript s to restrict these sets to variables or symbols of sort s.

[6] The equivalent of *if-then-else* constructions in functional programming.

Now, we assume given a (sorted) first-order term algebra $\mathcal{A} = T(\mathcal{H}, \mathcal{Z})$, called the algebra of *size expressions*, built from a non-empty set $\mathcal{H}$ of *size symbols* of fixed arity and a set $\mathcal{Z}$ of *size variables*. We assume that $\mathcal{H} \cap \mathcal{F} = \mathcal{Z} \cap \mathcal{X} = \emptyset$. Let $\mathcal{V}(t)$ be the set of size variables occurring in a term t. A *renaming* is an injection from a finite subset of $\mathcal{Z}$ to $\mathcal{Z}$.

We assume that, for every rule $l \to r \in \mathcal{R}$, $\mathcal{V}(l) = \mathcal{V}(r) = \emptyset$. Hence, if $t \to t'$ then, for all size substitution φ, $t\varphi \to t'\varphi$.

We also assume that $\mathcal{A}$ is equipped with a quasi-ordering $\leq_{\mathcal{A}}$ stable by size substitution (*i.e.* if $a \leq_{\mathcal{A}} b$ then, for all size substitution φ, $a\varphi \leq_{\mathcal{A}} b\varphi$) such that $(\mathcal{A}, \leq_{\mathcal{A}})$ has a well-founded model $(\mathfrak{A}, \leq_{\mathfrak{A}})$:

Definition 1 (Size model) A *pre-model* of $\mathcal{A}$ is given by a set $\mathfrak{A}$, an ordering $\leq_{\mathfrak{A}}$ on $\mathfrak{A}$ and a function $h_{\mathfrak{A}}$ from $\mathfrak{A}^n$ to $\mathfrak{A}$ for every n-ary size symbol $h \in \mathcal{H}$. A *size valuation* is a function ν from $\mathcal{Z}$ to $\mathfrak{A}$, naturally extended to a function on $\mathcal{A}$. A pre-model is a *model* if, for all size valuation ν, $a\nu \leq_{\mathfrak{A}} b\nu$ whenever $a \leq_{\mathcal{A}} b$. Such a model is *well-founded* if $>_{\mathfrak{A}}$ is well-founded.

The Calculus of Algebraic Constructions with Size Annotations (CACSA) is an extension of CAC where constant predicate symbols are annotated by size expressions. The terms of CACSA are defined by the following grammar rule:

$$t ::= s \mid x \mid C^a \mid f \mid [x : t]t \mid (x : t)t \mid tt$$

where $C \in \mathcal{CF}^{\square}$, $f \in \mathcal{F} \setminus \mathcal{CF}^{\square}$ and $a \in \mathcal{A}$. We denote by $\mathcal{T}_{\mathcal{A}}(\mathcal{F}, \mathcal{X})$ the set of terms built from $\mathcal{F}$, $\mathcal{X}$ and $\mathcal{A}$. Let $\underline{\mathcal{T}}$ be the set of the underlying CAC terms and $\_$ be the function erasing size annotations. Among CAC terms, we distinguish the following disjoint sets:

- *kinds*: $K \in \mathcal{K} ::= \star \mid (x : t)K$
- *predicates*: $P \in \mathcal{P} ::= f \in \mathcal{F}^{\square} \mid x \in \mathcal{X}^{\square} \mid (x : t)P \mid [x : t]P \mid Pt$
- *objects*: $o \in \mathcal{O} ::= f \in \mathcal{F}^{\star} \mid x \in \mathcal{X}^{\star} \mid [x : t]o \mid ot$

where $t \in \underline{\mathcal{T}}$ is any CAC term.

Finally, we assume that every symbol f is equipped with a type $\tau_f = (\vec{x} : \vec{T})U \in \mathcal{T}$ such that $\mathrm{FV}(\tau_f) = \emptyset$, $s_f = \square \Rightarrow \mathcal{V}(\tau_f) = \emptyset$, and $f\vec{l} \to r \in \mathcal{R} \Rightarrow |\vec{l}| \leq |\vec{t}|$.

We also assume that every symbol f is equipped with a set $\mathrm{Mon}^+(f) \subseteq A_f = \{1, \ldots, |\vec{x}|\}$ of *monotone arguments* and a set $\mathrm{Mon}^-(f) \subseteq A_f$ of *anti-monotone arguments* such that $\mathrm{Mon}^+(f) \cap \mathrm{Mon}^-(f) = \emptyset$. For a size symbol h, $\mathrm{Mon}^+(h)$ (resp. $\mathrm{Mon}^-(h)$) is taken to be the arguments in which $h_{\mathfrak{A}}$ is monotone (resp. anti-monotone).

An *environment* Γ is a sequence of pairs variable-term. Let $t \downarrow u$ iff there is v such that $t \to^* v \,{}^*\!\leftarrow u$. The typing rules of CACSA are given in Figure 4 and its subtyping rules in Figure 3. W.l.o.g. we can assume that, for all f, $\vdash \tau_f : s_f$. We also assume that, for every rule $l \to r \in \mathcal{R}$, there exist an environment Γ and a type T such that $\Gamma \vdash r : T$. This is to make sure that r is not ill-formed (see Lemma 12 in [11]).

Since, in the (symb) rule, symbol types are applied to arbitrary size substitutions φ, the name of size variables in symbol types is not relevant (size variables in symbol types are implicitly universally quantified).

A substitution θ *preserves typing* between Γ and Δ, written $\theta : \Gamma \rightsquigarrow \Delta$, iff $\Delta \vdash x\theta : x\Gamma\theta$ for all $x \in \mathrm{dom}(\Gamma)$. A type-preserving substitution satisfies the following important substitution property: if $\Gamma \vdash t : T$ and $\theta : \Gamma \rightsquigarrow \Delta$ then $\Delta \vdash t\theta : T\theta$.

Figure 3: Subtyping rules

$$\text{(refl)} \qquad T \leq T$$

$$\text{(size)} \qquad C^a\vec{t} \leq C^b\vec{t} \qquad (C \in \mathcal{CF}^{\Box},\ a \leq_{\mathcal{A}} b)$$

$$\text{(prod)} \qquad \frac{U' \leq U \quad V \leq V'}{(x:U)V \leq (x:U')V'}$$

$$\text{(conv)} \qquad \frac{T' \leq U'}{T \leq U} \qquad (T \downarrow T',\ U' \downarrow U)$$

$$\text{(trans)} \qquad \frac{T \leq U \quad U \leq V}{T \leq V}$$

In this paper, we make two important assumptions.

Assumptions:

(1) $\beta \cup \mathcal{R}$ is confluent. This is the case for instance if $\mathcal{R}$ is confluent and left-linear. Finding other sufficient conditions when there are type-level rewrite rules is an open problem.

(2) $\mathcal{R}$ preserves typing: if $l \to r \in \mathcal{R}$ and $\Gamma \vdash l\sigma : T$ then $\Gamma \vdash r\sigma : T$. Finding sufficient conditions with subtyping and dependent types does not seem easy as shown by the following example. We leave the study of this problem for future work.

Example 1 (Subject reduction) Assume that $s \in \mathcal{H}$, $nat : \star$, $s : nat^{\alpha} \Rightarrow nat^{s\alpha}$, $- : nat^{\alpha} \Rightarrow nat^{\beta} \Rightarrow nat^{\alpha}$, and let us prove that the rule $-(sx)(sy) \to -xy$ preserves typing. Assume that $\Gamma \vdash -(st)(su) : T$. We must prove that $\Gamma \vdash -tu : T$. By inversion, $\Gamma \vdash -(st) : (z_2 : T_2)U_2$, $\Gamma \vdash su : T_2$ and $U_2\{z_2 \mapsto su\} \leq T$. By inversion again, $\Gamma \vdash - : (z_1 : T_1)U_1$, $\Gamma \vdash st : T_1$ and $U_1\{z_1 \mapsto st\} \leq (z_2 : T_2)U_2$. Again, $nat^a \Rightarrow nat^b \Rightarrow nat^a \leq (z_1 : T_1)U_1$, $\Gamma \vdash s : (z_3 : T_3)U_3$, $\Gamma \vdash t : T_3$, $U_3\{z_3 \mapsto t\} \leq T_1$, $nat^c \Rightarrow nat^{sc} \leq (z_3 : T_3)U_3$, $\Gamma \vdash s : (z_4 : T_4)U_4$, $\Gamma \vdash u : T_4$, $U_4\{z_4 \mapsto u\} \leq T_2$ and $nat^d \Rightarrow nat^{sd} \leq (z_4 : T_4)U_4$. By Lemma 4, we have $T_3 \leq nat^c$, $nat^{sc} \leq U_3$, $T_4 \leq nat^d$, $nat^{sd} \leq U_4$, $T_1 \leq nat^a$ and $nat^b \Rightarrow nat^a \leq U_1$. Again, since $U_1\{z_1 \mapsto st\} \leq (z_2 : T_2)U_2$, $T_2 \leq nat^b$ and $nat^a \leq U_2$. Therefore, since $\Gamma \vdash t : T_3 \leq nat^c$, $\Gamma \vdash u : T_4 \leq nat^d$ and

Figure 4: Typing rules

$$\text{(ax)} \qquad \vdash \star : \Box$$

$$\text{(size)} \qquad \frac{\vdash \tau_C : \Box}{\vdash C^a : \tau_C} \qquad (C \in \mathcal{CF}^{\Box})$$

$$\text{(symb)} \qquad \frac{\vdash \tau_f : s_f}{\vdash f : \tau_f \varphi} \qquad (f \notin \mathcal{CF}^{\Box})$$

$$\text{(var)} \qquad \frac{\Gamma \vdash T : s_x}{\Gamma, x : T \vdash x : T} \qquad (x \notin \mathrm{dom}(\Gamma))$$

$$\text{(weak)} \qquad \frac{\Gamma \vdash t : T \quad \Gamma \vdash U : s_x}{\Gamma, x : U \vdash t : T} \qquad (x \notin \mathrm{dom}(\Gamma))$$

$$\text{(prod)} \qquad \frac{\Gamma \vdash U : s \quad \Gamma, x : U \vdash V : s'}{\Gamma \vdash (x : U)V : s'}$$

$$\text{(abs)} \qquad \frac{\Gamma, x : U \vdash v : V \quad \Gamma \vdash (x : U)V : s}{\Gamma \vdash [x : U]v : (x : U)V}$$

$$\text{(app)} \qquad \frac{\Gamma \vdash t : (x : U)V \quad \Gamma \vdash u : U}{\Gamma \vdash tu : V\{x \mapsto u\}}$$

$$\text{(sub)} \qquad \frac{\Gamma \vdash t : T \quad \Gamma \vdash T' : s}{\Gamma \vdash t : T'} \qquad (T \leq T')$$

$\Gamma \vdash - : nat^c \Rightarrow nat^d \Rightarrow nat^c$, we have $\Gamma \vdash -tu : nat^c$. Now, we must prove that $nat^c \leq T$. First, $nat^c \leq nat^{sc} \leq U_3$. Since $U_3\{z_3 \mapsto t\} \leq T_1$, $nat^c \leq T_1$. Since $nat^a \Rightarrow nat^b \Rightarrow nat^a \leq (z_1 : T_1)U_1$, $T_1 \leq nat^a$ and $nat^b \Rightarrow nat^a \leq U_1$. Since $U_1\{z_1 \mapsto st\} \leq (z_2 : T_2)U_2$, $nat^b \Rightarrow nat^a \leq (z_2 : T_2)U_2$. Therefore, $nat^a \leq U_2$. Now, since $U_2\{z_2 \mapsto su\} \leq T$, we indeed have $nat^c \leq T$.

3 Properties of subtyping

Lemma 2 If $U \leq V$ then, for all size substitution ψ, $U\psi \leq V\psi$.

Proof. Easy induction. ■

We now prove that the subtyping rule (trans) can be eliminated.

Theorem 3 (Transitivity elimination) Let $\leq_t$ be the subtyping relation obtained without using (trans). Then, $\leq_t = \leq$.

Proof. Section 9. ■

This means that, in a subtyping derivation, we can always assume that there is no application of (trans) and that, in a typing derivation, there is no successive applications of (sub).

Lemma 4 (Product compatibility) If $(x : U)V \leq (x : U')V'$ then $U' \leq U$ and $V \leq V'$.

Proof. By case on the last rule of $(x : U)V \leq (x : U')V'$. By confluence, we can assume that there is no successive applications of (conv). This is immediate for (refl) and (prod). (symb) is not possible. For (conv), we have:

$$\frac{(x : U)V \downarrow T \leq T' \downarrow (x : U')V'}{(x : U)V \leq (x : U')V'}$$

Then, we reason by case on the last rule of $T \leq T'$.

(refl) In this case, $T = T'$. Therefore, by confluence, $(x : U)V \downarrow (x : U')V'$, $U \downarrow U'$ and $V \downarrow V'$. Thus, $U' \leq U$ and $V \leq V'$.

(symb) Not possible since $T = C^a\vec{t}$ has no common reduct with $(x : U)V$ (since C is constant).

(conv) Excluded.

(prod) In this case, $T = (x : U_1)V_1$, $T' = (x : U_2)V_2$, $U_2 \leq U_1$ and $V_1 \leq V_2$. By confluence $U \downarrow U_1$, $V \downarrow V_1$, $U_2 \downarrow U'$ and $V_2 \downarrow V'$. Therefore, by conversion, $U' \leq U$ and $V \leq V'$. ■

We now prove that the subtyping relation can be further simplified. Consider the following two admissible rules:

$$\text{(red)} \quad \frac{T \rightarrow^* T' \quad T' \leq U' \quad U' \ {}^*\!\leftarrow U}{T \leq U}$$

$$\text{(exp)} \quad \frac{T \ {}^*\!\leftarrow T' \quad T' \leq U' \quad U' \rightarrow^* U}{T \leq U}$$

(conv) can clearly be replaced by both (red) and (exp).

Theorem 5 (Expansion elimination) Let $\leq_r$ be the subtyping relation with (red) instead of (conv). Then, $\leq_r = \leq$.

Proof. Section 10. ■

Now, let $\leq_s$ be the subtyping relation with (refl), (symb) and (prod) only.

Lemma 6 $T \leq U$ iff there exist T' and U' such that $T \rightarrow^* T' \leq_s U' \ {}^*\!\leftarrow U$. Furthermore, if $T, U \in \mathcal{WN}$ then $T\downarrow \leq_s U\downarrow$.

Proof. The if-part is immediate. The only-if-part is easily proved by induction on $T \leq U$. In the (red) case, if $T \to^* T' \leq U' \ {}^*\!\leftarrow U$ then, by induction hypothesis, there exist T'' and U'' such that $T' \to^* T'' \leq_s U'' \ {}^*\!\leftarrow U'$. Therefore, $T \to^* T'' \leq_s U'' \ {}^*\!\leftarrow U$.

Now, if $T, U \in \mathcal{WN}$ then $T{\downarrow} \leq U{\downarrow}$. Thus, $T{\downarrow} \leq_s U{\downarrow}$ since $T{\downarrow}$ and $U{\downarrow}$ are not reducible. ■

Lemma 7 – For all $s \in \mathcal{S}$, if $T \leq s$ or $s \leq T$ then $T \to^* s$.
– For all $K \in \mathcal{K}$, if $T \leq K$ or $K \leq T$ then $T \to^* T' \in \mathcal{K}$.

Proof.
– If $s \leq T$ then $s \leq_s T' \ {}^*\!\leftarrow T$. The only possible case is $T' = s$. If $T \leq s$ then $T \to^* T' \leq_s s$. The only possible case is $T' = s$.
– If $T \leq K$ then $T \to^* T' \leq_s K' \ {}^*\!\leftarrow K$ and $K' \in \mathcal{K}$. Now, one can easily prove by induction that, if $T' \leq_s K'$, then $T' \in \mathcal{K}$. If $K \leq T$ then $K \to^* K' \leq_s T' \ {}^*\!\leftarrow T$ and $K' \in \mathcal{K}$. One can easily prove by induction that, if $K' \leq_s T'$, then $T' \in \mathcal{K}$. ■

Theorem 8 (Decidability of subtyping) $\leq$ is decidable whenever $\to$ is confluent, weakly normalizing and finitely branching (or confluent and strongly normalizing).

Proof. Immediate consequence of Lemma 6.

4 Properties of typing

Lemma 9 If $\Gamma \vdash t : T$ then, for all size substitution ψ, $\Gamma\psi \vdash t\psi : T\psi$.

Proof. Easy induction. ■

Lemma 10 (Type correctness) If $\Gamma \vdash t : T$ then either $T = \Box$ or $\Gamma \vdash T : s$ for some sort s.

Proof. Easy induction. ■

Lemma 11 – If $T \to^* \Box$ then T is not typable.
– If $\Gamma \vdash t : \Box$ then $t \in \mathcal{K}$.
– If $K \in \mathcal{K}$ and $\Gamma \vdash K : L$ then $L = \Box$.
– If $T \to^* K \in \mathcal{K}$ and $\Gamma \vdash T : s$ then $T \in \mathcal{K}$ and $s = \Box$.

Proof. These properties are proved for CAC in [11] (Lemma 11). Their proofs need only a few corrections based on Lemma 7 to be valid for CACSA too. ■

Lemma 12 (Narrowing) If $\Gamma, y : A, \Gamma' \vdash t : T$, $B \leq A$, $\Gamma \vdash B : s_y$ then $\Gamma, y : B, \Gamma' \vdash t : T$.

Proof. By induction on $\Gamma, y : A, \Gamma' \vdash t : T$. We only detail some cases.

(var) There are two cases. Assume that we have $\Gamma \vdash A : s_y$ and $\Gamma, y : A \vdash y : A$. Since $\Gamma \vdash B : s_y$, by (var), $\Gamma, y : B \vdash y : B$. Since $B \leq A$ and $\Gamma \vdash A : s_y$, by (sub), $\Gamma, y : B \vdash y : A$.
Assume now that we have $\Gamma, y : A, \Gamma' \vdash T : s_x$ and $\Gamma, y : A, \Gamma', x : T \vdash x : T$. By induction hypothesis, $\Gamma, y : B, \Gamma' \vdash T : s_x$. Thus, by (var), $\Gamma, y : B, \Gamma', x : T \vdash x : T$.

(weak) There are two cases. Assume that we have $\Gamma \vdash t : T$, $\Gamma \vdash A : s_y$ and $\Gamma, y : A \vdash t : T$. Since $\Gamma \vdash B : s_y$, by (weak), $\Gamma, y : B \vdash t : T$.
Assume now that we have $\Gamma, y : A, \Gamma' \vdash t : T$, $\Gamma, y : A, \Gamma' \vdash U : s_x$ and $\Gamma, y : A, \Gamma', x : U \vdash t : T$. By induction hypothesis, $\Gamma, y : B, \Gamma' \vdash t : T$ and $\Gamma, y : B, \Gamma' \vdash U : s_x$. Thus, by (weak), $\Gamma, y : B, \Gamma', x : U \vdash t : T$. ■

Theorem 13 (β-Subject reduction) If $\Gamma \vdash t : T$ and $t \to_\beta t'$ then $\Gamma \vdash t' : T$.

Proof. By induction on $\Gamma \vdash t : T$, we also prove that, if $\Gamma \to_\beta \Gamma'$, then $\Gamma' \vdash t : T$. We only detail the case of a β-head reduction. Assume that we have $\Gamma \vdash [x : U']v : (x : U)V$ and $\Gamma \vdash u : U$. We must prove that $\Gamma \vdash v\{x \mapsto u\} : V\{x \mapsto u\}$. By inversion, $\Gamma, x : U' \vdash v : V'$, $\Gamma \vdash (x : U')V' : s'$, $(x : U')V' \leq (x : U)V$ and $\Gamma \vdash (x : U)V : s$. By product compatibility, $U \leq U'$ and $V' \leq V$. By inversion, $\Gamma \vdash U : s_1$ and $\Gamma \vdash V' : s_2$. By narrowing and subtyping, $\Gamma, x : U \vdash v : V$. Therefore, by substitution, $\Gamma \vdash v\{x \mapsto u\} : V\{x \mapsto u\}$. ■

Lemma 14 If $\Gamma \vdash t : T$, $T \leq T'$ and $\Gamma \vdash T' : s'$ then $\Gamma \vdash T : s$ for some s.

Proof. By type correctness, either $T = \Box$ or $\Gamma \vdash T : s$ for some s. If $T = \Box$ then, by Lemma 7, $T' \to^* \Box$ and, by Lemma 11, T' cannot be typable. ■

Lemma 15 (Unicity of sorting) If $T \leq T'$, $\Gamma \vdash T : s$ and $\Gamma \vdash T' : s'$ then $s = s'$.

Proof. If $s = \Box$ then $T \in \mathcal{K}$. By Lemma 7, $T' \to^* K \in \mathcal{K}$. By Lemma 11, $T' \in \mathcal{K}$ and $s' = \Box$. By symmetry, if $s' = \Box$ then $s = \Box$. So, $s = \Box$ iff $s' = \Box$. Since $s, s' \in \mathcal{S} = \{\star, \Box\}$, $s = \star$ iff $s' = \star$. Therefore, $s = s'$. ■

5 Strong normalization

Let $\mathcal{SN}$ (resp. $\mathcal{WN}$) be the set of strongly (resp. weakly) normalizable terms, and $t\!\downarrow$ be the normal form of a term $t \in \mathcal{WN}$ ($\to$ is assumed confluent).

Definition 16 (Reducibility candidates) We assume given a set $\mathcal{CT}$ of *constructor terms.*[7] A term t is *neutral* if it is not an abstraction, not a constructor term, nor of the form $f\vec{t}$ with $f \in \mathcal{DF}$ and $|\vec{t}| < |\vec{l}|$ for some rule $f\vec{l} \to r \in \mathcal{R}$. We inductively define the set $\mathcal{R}_t$ of the interpretations for the terms of type t,

[7] $\mathcal{CT}$ is defined in Definition 26.

the ordering $\leq_t$ on $\mathcal{R}_t$, the element $\top_t \in \mathcal{R}_t$, and the functions $\bigwedge_t$ and $\bigvee_t$ from the powerset of $\mathcal{R}_t$ to $\mathcal{R}_t$ as follows. If $t \notin \mathcal{K} \cup \{\Box\}$ then:

- $\mathcal{R}_t = \{\emptyset\}$, $\leq_t = \subseteq$ and $\bigwedge_t(\Re) = \bigvee_t(\Re) = \top_t = \emptyset$.

Otherwise:

- $\mathcal{R}_s$ is the set of all the subsets R of $\mathcal{T}$ such that:
  - **(R1)** $R \subseteq \mathcal{SN}$ (strong normalization).
  - **(R2)** If $t \in R$ then $\rightarrow(t) \subseteq R$ (stability by reduction).
  - **(R3)** If t is neutral and $\rightarrow(t) \subseteq R$ then $t \in R$ (neutral terms).

  Furthermore, $\leq_s = \subseteq$, $\top_s = \mathcal{SN}$, $\bigvee_s(\Re) = \bigcup \Re$, $\bigwedge_s(\Re) = \bigcap \Re$ if $\Re \neq \emptyset$, and $\bigwedge_s(\emptyset) = \top_s$.
- $\mathcal{R}_{(x:U)K}$ is the set of functions R from $\mathcal{T} \times \mathcal{R}_U$ to $\mathcal{R}_K$ such that $R(u, S) = R(u', S)$ whenever $u \rightarrow u'$ or $\underline{u} = \underline{u'}$, $\top_{(x:U)K}(u, S) = \top_K$, $\bigwedge_{(x:U)K}(\Re)(u, S) = \bigwedge_K(\{R(u, S) \mid R \in \Re\})$, $\bigvee_{(x:U)K}(\Re)(u, S) = \bigvee_K(\{R(u, S) \mid R \in \Re\})$ and $R \leq_{(x:U)K} R'$ iff $R(u, S) \leq_K R'(u, S)$.

Let $(\vec{t}, \vec{S}) \leq_i (\vec{t'}, \vec{S'})$ iff $\vec{t} = \vec{t'}$, $S_i \leq S'_i$ and, for all $j \neq i$, $S_j = S'_j$. A function $R \in \mathcal{R}_{(\vec{x}:\vec{T})\star}$ is *monotone* (resp. *anti-monotone*) in its ith argument if $R(\vec{Q}) \leq R(\vec{Q'})$ whenever $\vec{Q} \leq_i \vec{Q'}$ (resp. $\vec{Q} \geq_i \vec{Q'}$). Let $\mathcal{R}^m_{\tau_f}$ be the set of functions $R \in \mathcal{R}_{\tau_f}$ such that R is monotone in all its arguments $i \in \mathrm{Mon}^+(f)$, and anti-monotone in all its arguments $i \in \mathrm{Mon}^-(f)$.

Lemma 17 $(\mathcal{R}_t, \leq_t)$ and $(\mathcal{R}^m_t, \leq_t)$ are complete lattices with $\top_t$ as their greatest element and $\bigwedge_t(\Re)$ as the greatest lower bound of $\Re$. Moreover:

- If $\Re$ is totally ordered then $\bigvee_t(\Re)$ is the lowest upper bound of $\Re$.
- For all $R \in \mathcal{R}_s$, $\mathcal{X} \subseteq R$.
- If $\Gamma \vdash t : T$ and $\theta : \Gamma \rightsquigarrow \Delta$ then $\mathcal{R}_{T\theta} = \mathcal{R}_T$.
- If $\Gamma \vdash t : T$ then $\mathcal{R}_{T\varphi} = \mathcal{R}_T$.
- The smallest element $\bot_s = \bigwedge_s(\mathcal{R}_s)$ only contains neutral terms.

Proof. The proof is similar to the one for CAC [11]. ■

Lemma 18 If $\Gamma \vdash T \leq T' : s$ then $\mathcal{R}_T = \mathcal{R}_{T'}$.

Proof. If $s = \star$ then $\mathcal{R}_T = \{\emptyset\} = \mathcal{R}_{T'}$. Assume now that $s = \Box$. We proceed by induction on $T \leq T'$.

(refl) Immediate.

(symb) Not possible.

(prod) $\mathcal{R}_{(x:U)V}$ is the set of functions from $\mathcal{T} \times \mathcal{R}_U$ to $\mathcal{R}_V$ that are invariant by reduction and size substitution. $\mathcal{R}_{(x:U')V'}$ is the set of functions from $\mathcal{T} \times \mathcal{R}_{U'}$ to $\mathcal{R}_{V'}$ that are invariant by reduction and size substitution. By induction hypothesis, $\mathcal{R}_U = \mathcal{R}_{U'}$ and $\mathcal{R}_V = \mathcal{R}_{V'}$. Therefore, $\mathcal{R}_{(x:U)V} = \mathcal{R}_{(x:U')V'}$.

(conv) By induction hypothesis, $\mathcal{R}_{T'} = \mathcal{R}_{U'}$. Since $\mathcal{R}_T = \mathcal{R}_{T'}$ and $\mathcal{R}_U = \mathcal{R}_{U'}$, we have $\mathcal{R}_T = \mathcal{R}_U$. ■

Definition 19 (Interpretation schema) A *candidate assignment* is a function ξ from $\mathcal{X}$ to $\bigcup\{\mathcal{R}_t \mid t \in \mathcal{T}\}$. A candidate assignment ξ *validates* an environment Γ or is a Γ-assignment, $\xi \models \Gamma$, if, for all $x \in \mathrm{dom}(\Gamma)$, $x\xi \in \mathcal{R}_{x\Gamma}$.

An *interpretation* for a symbol $C \in \mathcal{CF}^{\Box}$ is a monotone function I from $\mathfrak{A}$ to $\mathcal{R}^m_{\tau_f}$. An *interpretation* for a symbol $f \notin \mathcal{CF}^{\Box}$ is an element of $\mathcal{R}^m_{\tau_f}$. An *interpretation* for a set $\mathcal{G}$ of predicate symbols is a function which, to every symbol $g \in \mathcal{G}$, associates an interpretation for g.

The *interpretation* of t w.r.t. a candidate assignment ξ, an interpretation I for $\mathcal{F}$, a substitution θ and a valuation ν, $[\![t]\!]^{I,\nu}_{\xi,\theta}$, is defined by induction on t:

- $[\![t]\!]^{I,\nu}_{\xi,\theta} = \top_t$ if $t \in \mathcal{O} \cup \mathcal{S}$
- $[\![F]\!]^{I,\nu}_{\xi,\theta} = I_F$ if $F \in \mathcal{DF}^{\Box}$
- $[\![C^a]\!]^{I,\nu}_{\xi,\theta} = I^{a\nu}_C$ if $C \in \mathcal{CF}^{\Box}$
- $[\![x]\!]^{I,\nu}_{\xi,\theta} = x\xi$
- $[\![(x:U)V]\!]^{I,\nu}_{\xi,\theta} = \{t \in \mathcal{T} \mid \forall u \in [\![U]\!]^{I,\nu}_{\xi,\theta}, \forall S \in \mathcal{R}_U, tu \in [\![V]\!]^{I,\nu}_{\xi^S_x,\theta^u_x}\}$
- $[\![[x:U]v]\!]^{I,\nu}_{\xi,\theta}(u,S) = [\![v]\!]^{I,\nu}_{\xi^S_x,\theta^u_x}$
- $[\![tu]\!]^{I,\nu}_{\xi,\theta} = [\![t]\!]^{I,\nu}_{\xi,\theta}(u\theta, [\![u]\!]^{I,\nu}_{\xi,\theta})$

where $\theta^u_x = \theta \cup \{x \mapsto u\}$ and $\xi^S_x = \xi \cup \{x \mapsto S\}$.

Let I be an interpretation for $\mathcal{F}$. A symbol f is *computable* if, for all ν, $f \in [\![\tau_f]\!]^{I,\nu}$. A substitution θ is *adapted* to a Γ-assignment ξ and a valuation ν, $\xi, \theta \models_\nu \Gamma$, if $\mathrm{dom}(\theta) \subseteq \mathrm{dom}(\Gamma)$ and, for all $x \in \mathrm{dom}(\theta)$, $x\theta \in [\![x\Gamma]\!]^{I,\nu}_{\xi,\theta}$. The interpretation is *invariant by reduction* if, for all ν, ξ, θ and $t, t' \in \mathcal{WN}$, $[\![t]\!]^{I,\nu}_{\xi,\theta} = [\![t']\!]^{I,\nu}_{\xi,\theta}$ whenever $t \to t'$.

Lemma 20 – If $\Gamma \vdash t : T$ and $\xi \models \Gamma$ then $[\![t]\!]^{I,\nu}_{\xi,\theta} \in \mathcal{R}_T$.
– If $\theta \to \theta'$ or $\underline{\theta} = \underline{\theta'}$ then $[\![t]\!]^{I,\nu}_{\xi,\theta} = [\![t]\!]^{I,\nu}_{\xi,\theta'}$.

Proof. The proof is similar to the one for CAC [11]. ■

Lemma 21 (Candidate substitution) If $\Gamma \vdash t : T$, $\gamma : \Gamma \rightsquigarrow \Delta$ and $\xi \models \Delta$ then $[\![t\gamma]\!]^{I,\nu}_{\xi,\sigma} = [\![t]\!]^{I,\nu}_{\eta,\gamma\sigma}$ with $x\eta = [\![x\gamma]\!]^{I,\nu}_{\xi,\sigma}$ and $\eta \models \Gamma$.

Proof. The proof is similar to the one for CAC [11]. ■

Lemma 22 (Size substitution) If $\Gamma \vdash t : T$ then $[\![t\varphi]\!]^{I,\nu}_{\xi,\theta} = [\![t]\!]^{I,\varphi\nu}_{\xi,\theta}$ where $\alpha(\varphi\nu) = (\alpha\varphi)\nu$.

Proof. By induction on t.
– If t is an object, a sort or a symbol $f \in \mathcal{F}^\star$ then $t\varphi$ is of the same kind and $[\![t\varphi]\!]^{I,\nu}_{\xi,\theta} = [\![t\varphi]\!]^{I,\nu}_{\xi,\theta} = \top_t$.
– $[\![C^a\varphi]\!]^{I,\nu}_{\xi,\theta} = I^{a\varphi\nu}_C = [\![C^a]\!]^{I,\varphi\nu}_{\xi,\theta}$.
– $[\![x\varphi]\!]^{I,\nu}_{\xi,\theta} = [\![x]\!]^{I,\nu}_{\xi,\theta} = x\xi$.

– $[\![(x:U\varphi)V\varphi]\!]^{I,\nu}_{\xi,\theta} = \{t \in \mathcal{T} \mid \forall u \in [\![U\varphi]\!]^{I,\nu}_{\xi,\theta}, \forall S \in \mathcal{R}_{U\varphi}, tu \in [\![V\varphi]\!]^{I,\nu}_{\xi^S_x,\theta^u_x}\}$. By induction hypothesis, $[\![U\varphi]\!]^{I,\nu}_{\xi,\theta} = [\![U]\!]^{I,\varphi\nu}_{\xi,\theta}$ and $[\![V\varphi]\!]^{I,\nu}_{\xi^S_x,\theta^u_x} = [\![V]\!]^{I,\varphi\nu}_{\xi^S_x,\theta^u_x}$. And since $\mathcal{R}_{U\varphi} = \mathcal{R}_U$, $[\![(x:U\varphi)V\varphi]\!]^{I,\nu}_{\xi,\theta} = [\![(x:U)V]\!]^{I,\nu}_{\xi,\theta}$.
– If $\Gamma \vdash [x:U]v : T$ then, by inversion, $\Gamma \vdash [x:U]v : (x:U)V$ for some V, and $\Gamma\varphi \vdash [x:U\varphi]v\varphi : (x:U\varphi)V\varphi$. Since $\mathcal{R}_{U\varphi} = \mathcal{R}_U$ and $\mathcal{R}_{V\varphi} = \mathcal{R}_V$, $[\![[x:U\varphi]v\varphi]\!]^{I,\nu}_{\xi,\theta}$ has the same domain and codomain as $[\![[x:U]v]\!]^{I,\nu}_{\xi,\theta}$. Furthermore, $[\![[x:U\varphi]v\varphi]\!]^{I,\nu}_{\xi,\theta}(u,S) = [\![v\varphi]\!]^{I,\nu}_{\xi^S_x,\theta^u_x} = [\![v]\!]^{I,\nu}_{\xi^S_x,\theta^u_x}$ by induction hypothesis.
– $[\![t\varphi u\varphi]\!]^{I,\nu}_{\xi,\theta} = [\![t\varphi]\!]^{I,\nu}_{\xi,\theta}(u\varphi\theta, [\![u\varphi]\!]^{I,\nu}_{\xi,\theta}) = [\![t]\!]^{I,\varphi\nu}_{\xi,\theta}(u\theta, [\![u]\!]^{I,\varphi\nu}_{\xi,\theta})$ by induction hypothesis and invariance by size change. ■

We now define the sets of positive and negative positions in a term, which will enforce monotony and anti-monotony properties respectively.

Definition 23 (Positive and negative positions) The set of positions in a term t is inductively defined as follows:[8]

– $\mathrm{Pos}(s) = \mathrm{Pos}(x) = \mathrm{Pos}(f) = \{\varepsilon\}$
– $\mathrm{Pos}((x:u)v) = \mathrm{Pos}([x:u]v) = \mathrm{Pos}(uv) = 1.\mathrm{Pos}(u) \cup 2.\mathrm{Pos}(v)$
– $\mathrm{Pos}(C^a) = \{\varepsilon\} \cup 0.\mathrm{Pos}(a)$

Let $\mathrm{Pos}(x,t)$ be the set of positions of the free occurrences of x in t, and $\mathrm{Pos}(f,t)$ be the set of positions of the occurrences of f in t. The set of *positive positions* in t, $\mathrm{Pos}^+(t)$, and the set of *negative positions* in t, $\mathrm{Pos}^-(t)$, are simultaneously defined by induction on t:

– $\mathrm{Pos}^\delta(s) = \mathrm{Pos}^\delta(x) = \{\varepsilon \mid \delta = +\}$
– $\mathrm{Pos}^\delta((x:U)V) = 1.\mathrm{Pos}^{-\delta}(U) \cup 2.\mathrm{Pos}^\delta(V)$
– $\mathrm{Pos}^\delta([x:U]v) = 2.\mathrm{Pos}^\delta(v)$
– $\mathrm{Pos}^\delta(tu) = 1.\mathrm{Pos}^\delta(t)$ if $t \neq f\vec{t}$
– $\mathrm{Pos}^\delta(f\vec{t}) = \{1^{|\vec{t}|} \mid \delta = +\} \cup \bigcup\{1^{|\vec{t}|-i}2.\mathrm{Pos}^{\varepsilon\delta}(t_i) \mid \varepsilon \in \{-,+\}, i \in \mathrm{Mon}^\varepsilon(f)\}$
– $\mathrm{Pos}^\delta(C^a\vec{t}) = \mathrm{Pos}^\delta(C\vec{t}) \cup \{1^{|\vec{t}|}0 \mid \delta = +\}.\mathrm{Pos}^\delta(a)$.

where $\delta \in \{-,+\}$, $-+ = -$ and $-- = +$ (usual rule of signs).

Lemma 24 (Monotony) Let $\leq^+ = \leq$; $\leq^- = \geq$; $\xi \leq_x \xi'$ iff $x\xi \leq x\xi'$ and, for all $y \neq x$, $y\xi = y\xi'$; $I \leq_f I'$ iff $I_f \leq I'_f$ and, for all $g \neq f$, $I_g = I'_g$; $\nu \leq_\alpha \nu'$ iff $\alpha\nu \leq_{\mathfrak{A}} \alpha\nu'$ and, for all $\beta \neq \alpha$, $\beta\nu = \beta\nu'$. Assume that $\Gamma \vdash t : T$ and $\xi, \xi' \models \Gamma$.

– If $\xi \leq_x \xi'$ and $\mathrm{Pos}(x,t) \subseteq \mathrm{Pos}^\delta(t)$ then $[\![t]\!]^{I,\nu}_{\xi,\theta} \leq^\delta [\![t]\!]^{I,\nu}_{\xi',\theta}$.
– If $I \leq_f I'$ and $\mathrm{Pos}(f,t) \subseteq \mathrm{Pos}^\delta(t)$ then $[\![t]\!]^{I,\nu}_{\xi,\theta} \leq^\delta [\![t]\!]^{I',\nu}_{\xi,\theta}$.
– If $\nu \leq_\alpha \nu'$ and $\mathrm{Pos}(\alpha,t) \subseteq \mathrm{Pos}^\delta(t)$ then $[\![t]\!]^{I,\nu}_{\xi,\theta} \leq^\delta [\![t]\!]^{I,\nu'}_{\xi,\theta}$.
– If $\Gamma \vdash T \leq T' : s, T, T' \in \mathcal{WN}$ and the interpretation is invariant by reduction then $[\![T]\!]^{I,\nu}_{\xi,\theta} \leq [\![T']\!]^{I,\nu}_{\xi,\theta}$.

[8] It is defined so that $\mathrm{Pos}(\underline{t}) \subseteq \mathrm{Pos}(t)$.

Proof.

- The first two properties are proved for CAC in [11] and their proofs are still valid.
- We now prove the third property. It uses the same techniques. So, we only detail the case $t = C^a\vec{t}$. Let $R = [\![t]\!]^{I,\nu}_{\xi,\theta}$ and $R' = [\![t]\!]^{I,\nu'}_{\xi,\theta}$. $R = I^{a\nu}_C(\vec{t\theta}, \vec{S})$ with $\vec{S} = [\![\vec{t}]\!]^{I,\nu}_{\xi,\theta}$, and $R' = I^{a\nu'}_C(\vec{t\theta}, \vec{S'})$ with $\vec{S} = [\![\vec{t}]\!]^{I,\nu'}_{\xi,\theta}$. Let $n = |\vec{t}|$ and $i \in \{1, \ldots, n\}$. If $\mathrm{Pos}(\alpha, t_i) = \emptyset$ then $S_i = S'_i$. Otherwise, since $\mathrm{Pos}(\alpha, t) \subseteq \mathrm{Pos}^\delta(t)$, there is ε_i such that $i \in \mathrm{Mon}^{\varepsilon_i}(f)$ and $\mathrm{Pos}(\alpha, t_i) \subseteq \mathrm{Pos}^{\varepsilon_i\delta}(t_i)$. Thus, by induction hypothesis, $S_i \leq^{\varepsilon_i\delta} S'_i$. Let $Q^k_j = (\vec{t\theta}, S'_j)$ if $j \leq k$, and $Q^k_j = (\vec{t\theta}, S_j)$ if $j > k$. We have $\vec{Q}^0 = (\vec{t\theta}, \vec{S})$, $\vec{Q}^n = (\vec{t\theta}, \vec{S'})$ and, for all $k \in \{1, \ldots, n\}$, $\vec{Q}^{k-1} \leq^{\varepsilon_k\delta}_k \vec{Q}^k$. Thus, $I^{a\nu}_C(\vec{Q}^{k-1}) \leq^{\varepsilon^2_k\delta} I^{a\nu}_C(\vec{Q}^k)$, that is, $I^{a\nu}_C(\vec{Q}^{k-1}) \leq^\delta I^{a\nu}_C(\vec{Q}^k)$ since $\varepsilon^2_k = +$ and symbol interpretations are monotone in their monotone arguments and anti-monotone in their anti-monotone arguments. So, $R = I^{a\nu}_C(\vec{Q}^0) \leq^\delta I^{a\nu}_C(\vec{Q}^n)$. Now, if $\mathrm{Pos}(\alpha, C^a) = \emptyset$ then $a\nu = a\nu'$ and $R \leq^\delta R' = I^{a\nu}_C(\vec{Q}^n)$. Otherwise, $\delta = +$ and $a\nu \leq_{\mathfrak{A}} a\nu'$ since $\mathrm{Pos}(\alpha, a) \subseteq \mathrm{Pos}^+(a)$. Thus, $R \leq R'$ since symbol interpretations are monotone functions on $\mathfrak{A}$.
- We now prove the last property by induction on $T \leq T'$. Let $R = [\![T]\!]^{I,\nu}_{\xi,\theta}$ and $R' = [\![T']\!]^{I,\nu}_{\xi,\theta}$,

  (refl) Immediate.

  (symb) Let $\vec{Q} = (\vec{t\theta}, [\![\vec{t}]\!]^{I,\nu}_{\xi,\theta})$. We have $R = I^{a\nu}_C(\vec{Q}) \leq R' = I^{b\nu}_C(\vec{Q})$ since $a\nu \leq_{\mathfrak{A}} b\nu$ and symbol interpretations are monotone on $\mathfrak{A}$.

  (prod) Let $t \in R$, $u \in [\![U']\!]^{I,\nu}_{\xi,\theta}$ and $S \in \mathcal{R}_{U'}$. We must prove that $tu \in [\![V']\!]^{I,\nu}_{\xi^S_x,\theta^u_x}$. By induction hypothesis, $[\![U']\!]^{I,\nu}_{\xi,\theta} \leq [\![U]\!]^{I,\nu}_{\xi,\theta}$. So, $u \in [\![U]\!]^{I,\nu}_{\xi,\theta}$. Since $\mathcal{R}_{U'} = \mathcal{R}_U$ and $t \in R$, $tu \in [\![V]\!]^{I,\nu}_{\xi^S_x,\theta^u_x}$. Now, by induction hypothesis, $[\![V]\!]^{I,\nu}_{\xi^S_x,\theta^u_x} \leq [\![V']\!]^{I,\nu}_{\xi^S_x,\theta^u_x}$. Therefore, $tu \in [\![V']\!]^{I,\nu}_{\xi^S_x,\theta^u_x}$.

  (conv) By induction hypothesis, $[\![T']\!]^{I,\nu}_{\xi,\theta} \leq [\![U']\!]^{I,\nu}_{\xi,\theta}$. Since $T, U \in \mathcal{WN}$ and the interpretation is invariant by reduction, $[\![T']\!]^{I,\nu}_{\xi,\theta} = R$ and $[\![U']\!]^{I,\nu}_{\xi,\theta} = R'$. Therefore, $R \leq R'$. ■

Theorem 25 (Strong normalization) If there is an interpretation I invariant by reduction and such that every symbol is computable then every well-typed term is strongly normalizable.

Proof. One first prove by induction that, if $\Gamma \vdash t : T$ then, for all ξ, ν and θ such that $\xi \models \Gamma$ and $\xi, \theta \models_\nu \Gamma$, then $t\theta \in [\![T]\!]^\nu_{\xi,\theta}$. Then, one prove that, if $x\theta = x$ and $x\xi = \top_{x\Gamma}$, then $\xi \models \Gamma$ and $\xi, \theta \models_\nu \Gamma$. See [11] for details. ■

6 Constructor-based systems

We now study the case of CACSA's whose size algebra contains the following expressions (at least):

$$a ::= \alpha \mid sa \mid \infty \mid \ \ldots$$

In case that there is no other symbol, the ordering $\leq_{\mathcal{A}}$ on size expressions is defined as the smallest quasi-ordering $\leq$ such that, for all a, $a < sa$ and $a \leq \infty$. We interpret size expressions in the set $\mathfrak{A} = \Omega + 1$, where Ω is the first uncountable ordinal, by taking:

- $s_{\mathfrak{A}}(\mathfrak{a}) = \mathfrak{a} + 1$ if $\mathfrak{a} < \Omega$, and Ω otherwise.
- $\infty_{\mathfrak{A}} = \Omega$.

One can easily imagine other size expressions like $a + b$, $max(a, b)$, ...

Definition 26 (Constructor-based system) We assume given a *precedence* $\leq_{\mathcal{F}}$ on $\mathcal{F}$, that is, a quasi-ordering whose strict part $>_{\mathcal{F}}$ is well-founded, and that every $C \in \mathcal{CF}^{\Box}$ with $C : (\vec{z} : \vec{V})\star$ is equipped with a set $\mathrm{Cons}(C)$ of *constructors*, that is, a set of constant symbols $f : (\vec{y} : \vec{U})C^a\vec{v}$ equipped with a set $\mathrm{Acc}(f) \subseteq \{1, \ldots, |\vec{y}|\}$ of *accessible* arguments such that:

- If there are $D =_{\mathcal{F}} C$ and $j \in \mathrm{Acc}(c)$ such that $\mathrm{Pos}(D, U_j) \neq \emptyset$ then $\mathcal{V}(\tau_f) = \{\alpha\}$ and $a = s\alpha$.
- For all $j \in \mathrm{Acc}(c)$:
  - For all $D >_{\mathcal{F}} C$, $\mathrm{Pos}(D, U_j) = \emptyset$.
  - For all $D \simeq_{\mathcal{F}} C$ and $p \in \mathrm{Pos}(D, U_j)$, $p \in \mathrm{Pos}^+(U_j)$ and $U_j|_p = D^{\alpha}$.
  - For all $p \in \mathrm{Pos}(\alpha, U_j)$, $p = q0$, $U_j|_q = D^{\alpha}$ and $D \simeq_{\mathcal{F}} C$.
  - For all $x \in \mathrm{FV}^{\Box}(U_j)$, there is ι_x with $v_{\iota_x} = x$ and $\mathrm{Pos}(x, U_j) \subseteq \mathrm{Pos}^+(U_j)$.
- For all $F \in \mathcal{DF}^{\Box}$ and $F\vec{l} \to r \in \mathcal{R}$:
  - For all $G >_{\mathcal{F}} F$, $\mathrm{Pos}(G, r) = \emptyset$.
  - For all $i \in \mathrm{Mon}^{\delta}(F)$, $l_i \in \mathcal{X}^{\Box}$ and $\mathrm{Pos}(l_i, r) \subseteq \mathrm{Pos}^{\delta}(r)$.
  - For all $x \in \mathrm{FV}^{\Box}(r)$, there is κ_x with $l_{k_x} = x$.

A *C-constructor term* is a term of the form $f\vec{u}$ with $f \in \mathrm{Cons}$, $f : (\vec{y} : \vec{U})C^a\vec{v}$, $|\vec{u}| = |\vec{y}|$ and $\mathrm{Acc}(f) \neq \emptyset$. Let $\mathcal{CT}(C)$ be the set of C-constructor terms.

The conditions involving ι_x and κ_x means that we restrict our attention to *small* inductive types. Strong elimination, that is, predicate-level recursion on big inductive types may lead to non-termination [18]. Yet, weak elimination, that is, object-level recursion on big inductive types is admissible. As shown in [8], it is possible to raise this restriction at the price of not being allowed to match defined symbols.

Among constant predicate symbols, we distinguish the class of primitive types that includes all first-order data type like natural numbers, lists of natural numbers, ... Primitive types are not polymophic but they can have primitive dependancies like the type of arrays of natural numbers.

Definition 27 (Primitive types) A symbol $C \in \mathcal{CF}^{\Box}$ is *primitive* if $\tau_C = (\vec{z} : \vec{V})\star$, $\{\vec{z}\} \subseteq \mathcal{X}^{\star}$ and, for all $D \simeq_{\mathcal{F}} C$, for all constructor $f : (\vec{y} : \vec{U})D^{s\alpha}\vec{v}$ and for all $j \in \mathrm{Acc}(f)$, either $U_j = E^{\infty}\vec{t}$ with $E <_{\mathcal{F}} C$ and E primitive, or $U_j = E^{\alpha}\vec{t}$ with $E \simeq_{\mathcal{F}} C$. The *size* of a term t in a primitive type C is defined

as follows. If t is a constructor term $f\vec{u}$ with $f : (\vec{y} : \vec{U})C^{s\alpha}\vec{v}$ and, for all $j \in \mathrm{Acc}(f)$ such that $\mathrm{Pos}(\alpha, U_j) \neq \emptyset$, $U_j = C_j^{\alpha}\vec{v}^j$, then $|t|_C = 1 + max\{|u_j|_{C_j} \mid j \in \mathrm{Acc}(f), \mathrm{Pos}(\alpha, U_j) \neq \emptyset\}$. Otherwise, $|t|_C = 0$.

We define the interpretation of predicate symbols by induction on $>_{\mathcal{F}}$.

Definition 28 (Interpretation of defined predicate symbols) Assume that $F : (\vec{x} : \vec{T})U$. We take $I_F(\vec{t}, \vec{S}) = [\![r]\!]^I_{\xi,\sigma}$ if $\vec{t} \in \mathcal{WN}$, $\vec{t}\downarrow = \vec{l}\sigma$, $F\vec{l} \to r \in \mathcal{R}$ and $x\xi = S_{\kappa_x}$. Otherwise, we take $I_F(\vec{t}, \vec{S}) = \top_U$.

Thanks to Lemma 24, one can easily check that I is monotone in its monotone arguments. The well-foundedness of the definition is a consequence of the correctness of the termination criterion.

We now define the interpretation of a constant predicate symbols by transfinite induction on $\mathfrak{a} \in \mathfrak{A}$.

Definition 29 (Interpretation of constant predicate symbols)

- $I_C^0(\vec{S})$[9] is the set of $u \in \mathcal{SN}$ such that u never reduces to a C-constructor term.
- $I_C^{\mathfrak{a}+1}(\vec{S})$ is the set of terms $u \in \mathcal{SN}$ such that, if u reduces to a constructor term $f\vec{u}$ with $f : (\vec{y} : \vec{U})C^{s\alpha}\vec{v}$ then, for all $j \in \mathrm{Acc}(f)$, $u_j \in [\![U_j]\!]^{I,\nu}_{\xi,\theta}$ with $y\xi = S_{\iota_y}$, $\vec{y}\theta = \vec{u}$ and $\alpha\nu = \mathfrak{a}$.
- $I_C^{\mathfrak{b}} = \bigwedge_{\tau_C}(\{I_C^{\mathfrak{a}} \mid \mathfrak{a} < \mathfrak{b}\})$ if $\mathfrak{b}$ is a limit ordinal.

Let $K_C^{\mathfrak{a}}(\vec{S}) = I_C^{\mathfrak{a}}(\vec{S}) \cap \mathcal{CT}(C)$ and, for $t \in I_C^{\Omega}(\vec{S})$, let $o_{C(\vec{S})}(t)$ be the smallest ordinal $\mathfrak{a}$ such that $t \in I_C^{\mathfrak{a}}(\vec{S})$.

The interpretation is well defined thanks to the assumptions made on U_j when j is accessible.

Lemma 30 If $f\vec{u} \in K_C^{\Omega}(\vec{S})$ then $o_{C(\vec{S})}(f\vec{u})$ is a successor ordinal.

Proof. Assume that $\mathfrak{a} = o_{C(\vec{S})}(f\vec{u})$ is a limit ordinal. Then, $I_C^{\mathfrak{a}}(\vec{S}) = \bigcup\{I_C^{\mathfrak{b}}(\vec{S}) \mid \mathfrak{b} < \mathfrak{a}\}$ and $t\sigma \in I_C^{\mathfrak{b}}(\vec{S})$ for some $\mathfrak{b} < \mathfrak{a}$, which is not possible. Now, $\mathfrak{a} \neq 0$ since $K_C^0(\vec{S}) = \emptyset$. Therefore, $\mathfrak{a}$ is a successor ordinal. ■

Lemma 31 I is monotone.

Proof. We prove that $\mathfrak{a} \leq \mathfrak{b} \Rightarrow I^{\mathfrak{a}} \leq I^{\mathfrak{b}}$ by induction on $\mathfrak{a}$.

- $\mathfrak{a} = 0$.
  - $\mathfrak{b} = 0$. Immediate.

[9] We do not write $\vec{t}$ since the interpretation does not depend on it.

- $\mathfrak{b} = \mathfrak{b}' + 1$. By induction hypothesis, $I^0 \leq I^{\mathfrak{b}'}$. We now prove that $I^{\mathfrak{b}'} \leq I^{\mathfrak{b}'+1}$. Let $t \in I_C^{\mathfrak{b}'}(\vec{S})$. Then, $t \in \mathcal{SN}$. Assume now that t reduces to a constructor term $f\vec{u}$ with $f : (\vec{y} : \vec{U})C^{s\alpha}\vec{v}$. By Lemma 30, $t \in I_C^{\mathfrak{c}+1}(\vec{S})$ for some $\mathfrak{c} < \mathfrak{b}'$. Let $j \in \mathrm{Acc}(f)$. Then, $u_j \in [\![U_j]\!]^{\nu}_{\xi,\theta}$ with $y\xi = S_{\iota_y}$, $\vec{y}\theta = \vec{u}$ and $\alpha\nu = \mathfrak{c}$. After the conditions on U_j, by Lemma 24, $[\![U_j]\!]^{\nu}_{\xi,\theta} \subseteq [\![U_j]\!]^{\mu}_{\xi,\theta}$ where $\alpha\mu = \mathfrak{b}'$. Thus, $t \in I_C^{\mathfrak{b}'+1}(\vec{S})$.
- $\mathfrak{b}$ is a limit ordinal. By induction hypothesis, $I^0 \leq I^{\mathfrak{b}'}$ for all $\mathfrak{b}' < \mathfrak{b}$. Thus, $I^0 \leq I^{\mathfrak{b}}$.

• $\mathfrak{a} = \mathfrak{a}' + 1$.
- $\mathfrak{b} = 0$. Not possible.
- $\mathfrak{b} = \mathfrak{b}' + 1$. Then, $\mathfrak{a}' \leq \mathfrak{b}'$. Let $t \in I_C^{\mathfrak{a}}(\vec{S})$. Then, $t \in \mathcal{SN}$. Assume now that t reduces to a constructor term $f\vec{u}$ with $f : (\vec{y} : \vec{U})C^{s\alpha}\vec{v}$ and let $j \in \mathrm{Acc}(f)$. Then, $u_j \in [\![U_j]\!]^{\nu}_{\xi,\theta}$ with $y\xi = S_{\iota_y}$, $\vec{y}\theta = \vec{u}$ and $\alpha\nu = \mathfrak{a}'$. After the conditions on U_j, by Lemma 24, $[\![U_j]\!]^{\nu}_{\xi,\theta} \subseteq [\![U_j]\!]^{\mu}_{\xi,\theta}$ where $\alpha\mu = \mathfrak{b}'$. Thus, $t \in I_C^{\mathfrak{b}}(\vec{S})$.
- $\mathfrak{b}$ is a limit ordinal. Then, $\mathfrak{a}' < \mathfrak{b}'$ for some $\mathfrak{b}' < \mathfrak{b}$ and we can conclude by induction hypothesis.

• $\mathfrak{a}$ is a limit ordinal.
- $\mathfrak{b} = 0$. Not possible.
- $\mathfrak{b} = \mathfrak{b}' + 1$. Then, $\mathfrak{a} \leq \mathfrak{b}'$ and we can conclude by induction hypothesis.
- $\mathfrak{b}$ is a limit ordinal. Then, for all $\mathfrak{a}' < \mathfrak{a}$, $\mathfrak{a}' < \mathfrak{b}$, and we can conclude by induction hypothesis. ■

Lemma 32 (Primitive types) Let C be primitive type. If $\mathfrak{a} \geq \omega$ then $I_C^{\mathfrak{a}} = \top_{\tau_C}$. Otherwise, $I_C^{\mathfrak{a}}(\vec{S}) = \{t \in \mathcal{SN} \mid |t{\downarrow}|_C \leq \mathfrak{a}\}$, that is, $o_{C(\vec{s})}(t) = |t{\downarrow}|_C$.

Proof. We proceed by induction on C with $>_{\mathcal{F}}$ as well-founded ordering.

Let $J_C^{\mathfrak{a}} = \{t \in \mathcal{SN} \mid |t{\downarrow}|_C \leq \mathfrak{a}\}$. Since primitive types are not polymorphic, every $S_i = \emptyset$. So, we can drop the arguments $\vec{S}$. Note also that $|t|_C \leq |t'|_C$ whenever $t \to t'$ (since $\mathrm{Cons} \subseteq \mathcal{CF}$).

We first prove that, for all $\mathfrak{a} < \omega$, if $o_C(t) = \mathfrak{a}$ then $|t{\downarrow}|_C = \mathfrak{a}$.
- $\mathfrak{a} = 0$. If $o_C(t) = 0$ then $t \in I_C^0 \subseteq J_C^0$. Thus, $|t{\downarrow}|_C = 0$.
- $\mathfrak{a} = \mathfrak{a}' + 1$. If $o_C(t) = \mathfrak{a}' + 1$ then $t \in I_C^{\mathfrak{a}'+1} \setminus I_C^{\mathfrak{a}'}$. Since $t \notin I_C^0$, t reduces to a constructor term $f\vec{u}$ with $f : (\vec{y} : \vec{U})C^{s\alpha}\vec{v}$. Let $j \in \mathrm{Acc}(f)$. Then, $u_j \in [\![U_j]\!]^{\nu}_{\xi,\theta}$ with $y\xi = S_{\iota_y}$, $\vec{y}\theta = \vec{u}$ and $\alpha\nu = \mathfrak{a}'$. Moreover, either $U_j = C_j^{\alpha}\vec{v}^j$ with $C_j \simeq_{\mathcal{F}} C$, or $U_j = C_j^{\infty}\vec{v}^j$ with $C_j <_{\mathcal{F}} C$. In the former case, $u_j \in I_{C_j}^{\mathfrak{a}'}$. Thus, $o_{C_j}(u_j) \leq \mathfrak{a}'$ and, by induction hypothesis, $o_{C_j}(u_j) = |u_j{\downarrow}|_{C_j}$. Therefore, $o_C(t) = |t{\downarrow}|$.

Thus $o_C(t) = |t{\downarrow}|_C$ and, for all $\mathfrak{a} < \omega$, $I_C^{\mathfrak{a}} = J_C^{\mathfrak{a}}$. We now prove that $I_C^{\omega+1} = I_C^{\omega} = \mathcal{SN}$. Let $t \in I_C^{\omega+1} \setminus I_C^{\omega}$. Since $t \notin I_C^0$, t reduces to a constructor term $f\vec{u}$ with $f : (\vec{y} : \vec{U})C^{s\alpha}\vec{v}$ and, for all $j \in \mathrm{Acc}(f)$, $u_j \in [\![U_j]\!]^{\nu}_{\xi,\theta}$ with $y\xi = S_{\iota_y}$, $\vec{y}\theta = \vec{u}$ and $\alpha\nu = \omega$. Thus, for all $j \in \mathrm{Acc}(f)$, there is $\mathfrak{a}_j < \omega$ such that $u_j \in [\![U_j]\!]^{\nu_j}_{\xi,\theta}$ with $\alpha\nu_j = \mathfrak{a}_j$. $\mathfrak{a} = max\{\mathfrak{a}_j \mid j \in \mathrm{Acc}(f)$ is well defined since

$\mathrm{Acc}(f) \neq \emptyset$ and $\mathfrak{a} < \omega$ since $\mathrm{Acc}(f)$ is finite. Thus, $t \in I_C^{\mathfrak{a}+1} \subseteq I_C^{\omega}$. ■

We now give general conditions for every symbol to be computable, based on the fundamental notion of *computability closure*. The computability closure of a term t is a set of terms that can be proved computable whenever t is computable. If, for every rule $f\vec{l} \to r$, r belongs to the computability closure of $\vec{l}$, then rules preserve computability, hence strong normalization.

In [11], the computability closure is inductively defined as a typing relation $\vdash_c$ similar to $\vdash$ except for the (symb) case which is replaced by two new cases: (symb$^<$) for symbols strictly smaller than f, and (symb$^=$) for symbols equivalent to f whose arguments are structurally smaller than $\vec{l}$.

Here, we propose to add a new case for symbols equivalent to f whose arguments have sizes strictly smaller than those of $\vec{l}$. For comparing the sizes, one can use metrics like in [42].

Definition 33 (Ordering on symbol arguments) For every symbol $f : (\vec{x} : \vec{T})U$, we assume given two well-founded domains, $(D_f^{\mathcal{A}}, >_f^{\mathcal{A}})$ and $(D_f^{\mathfrak{A}}, >_f^{\mathfrak{A}})$, and two measure/metric functions $\zeta_f^{\mathcal{A}} : \mathcal{A}^n \to D_f^{\mathcal{A}}$ and $\zeta_f^{\mathfrak{A}} : \mathfrak{A}^n \to D_f^{\mathfrak{A}}$ $(n = |\vec{x}|)$ such that $(D_f^X, >_f^X) = (D_g^X, >_f^X)$ $(X \in \{\mathcal{A}, \mathfrak{A}\})$ whenever $f \simeq_{\mathcal{F}} g$, and we define:

- $a_f^i = a$ if $T_i = C^a\vec{v}$, and $a_f^i = \infty$ otherwise.
- $(f, \varphi) >^{\mathcal{A}} (g, \psi)$ iff $f >_{\mathcal{F}} g$ or $f \simeq_{\mathcal{F}} g$ and $\zeta_f^{\mathcal{A}}(\vec{a}_f\varphi) >_f^{\mathcal{A}} \zeta_g^{\mathcal{A}}(\vec{a}_g\psi)$.
- $(f, \nu) >^{\mathfrak{A}} (g, \mu)$ iff $f >_{\mathcal{F}} g$ or $f \simeq_{\mathcal{F}} g$ and $\zeta_f^{\mathfrak{A}}(\vec{a}_f\nu) >_f^{\mathfrak{A}} \zeta_g^{\mathfrak{A}}(\vec{a}_g\mu)$.

Then, we assume that $>^{\mathcal{A}}$ is decidable and that (for all ν) $(f, \varphi\nu) >^{\mathfrak{A}} (g, \psi\nu)$ whenever $(f, \varphi) >^{\mathcal{A}} (g, \psi)$.

Example 2 (Lexicographic and multiset status) A simple metric is given by assigning a *status* to every symbol, that is, a non-empty sequence of finite multisets of strictly positive integers, describing a simple combination of lexicographic and multiset comparisons. Given a set D and a status ζ of arity n (biggest integer occurring in it), we define $[\![\zeta]\!]_D$ on D^n as follows:

- $[\![M_1 \dots M_k]\!]_D(\vec{x}) = ([\![M_1]\!]_D^m(\vec{x}), \dots, [\![M_k]\!]_D^m(\vec{x}))$
- $[\![\{i_1, \dots, i_p\}]\!]_D^m(\vec{x}) = \{x_{i_1}, \dots, x_{i_p}\}$ (multiset)

Now, take $\zeta_f^X = [\![\zeta_f]\!]_X$, $D_f^X = \zeta_f^X(X^n)$ and $>_f^X = ((>_X)_{\mathrm{mul}})_{\mathrm{lex}}$.

For building the computability closure, one must start from the variables of the left hand-side. However, one cannot take any variable since not every subterm of a computable term is computable *a priori*. To this end, based on the definition of the interpretation of constant predicate symbols, we introduce the notion of accessibility.

Definition 34 (Accessibility) We say that $u : U$ is *a-accessible*[10] in $t : T$, written $t : T \rhd_a u : U$, iff $t = f\vec{u}$, $f \in \mathrm{Cons}$, $f : (\vec{y} : \vec{U})C^{s\alpha}\vec{v}$, $|\vec{u}| = |\vec{y}|$,

[10] We may not indicate a if it is not relevant.

$u = u_j$, $j \in \mathrm{Acc}(f)$, $T = C^{s\alpha\varphi}\vec{v}\gamma$, $U = U_j\gamma\varphi$, $\gamma = \{\vec{y} \mapsto \vec{u}\}$, $\varphi = \{\alpha \mapsto a\}$ and $\mathrm{Pos}(\alpha, \vec{u}) = \emptyset$.

A constructor $c : (\vec{y} : \vec{U})C^a\vec{v}$ is *finitely branching*[11] iff, for all $j \in \mathrm{Acc}(c)$, either $\mathrm{Pos}(\alpha, U_j) = \emptyset$ or there exists D such that $U_j = D^\alpha\vec{u}$. We say that $u : U$ is *strongly a-accessible* in $t : T$, written $t : T \rhd_a u : U$, iff $t : T \rhd_a u : U$, f is a finitely branching constructor and $\mathrm{Pos}(\alpha, U_j) \neq \emptyset$.

We say that $u : U$ is *$*$-accessible modulo φ* in $t : T$, written $t : T \gg_\varphi u : U$, iff either $t : T\varphi = u : U$ and $\varphi|_{\mathcal{V}(T)}$ is a renaming, or $t : T\varphi \rhd^* \rhd_\epsilon u : U$ for some size variable ϵ.

Definition 35 (Termination criterion) Let $(f\vec{l} \to r, \Gamma, \varphi) \in \mathcal{R}$ with $f : (\vec{x} : \vec{T})U$ and $\gamma = \{\vec{x} \mapsto \vec{l}\}$. The *computability closure* associated to this rule is given by the type system of Figure 5 on the set of terms $\mathcal{T}_\mathcal{A}(\mathcal{F}', \mathcal{X}')$ where $\mathcal{F}' = \mathcal{F} \cup \mathrm{dom}(\Gamma)$, $\mathcal{X}' = \mathcal{X} \setminus \mathrm{dom}(\Gamma)$ and, for all $x \in \mathrm{dom}(\Gamma)$, $\tau_x = x\Gamma$ and $x <_\mathcal{F} f$. The termination conditions are:

- Well-typedness: for all $x \in \mathrm{dom}(\Gamma)$, $\vdash_c l_i : T_i\varphi\gamma$.
- Linearity: Γ is linear w.r.t. size variables.
- Accessibility: for all $x \in \mathrm{dom}(\Gamma)$, there are i and β such that $l_i : T_i\gamma \gg_\varphi x : x\Gamma$,[12] $T_i = C^\beta\vec{t}$ and $\mathcal{V}(\vec{t}) = \emptyset$.
- Computability closure: $\vdash_c r : U\varphi\gamma$.
- Positivity: for all $\alpha \in \mathcal{V}(\vec{T})$, $\mathrm{Pos}(\alpha, U) \subseteq \mathrm{Pos}^+(U)$.
- Safeness: γ is an injection from $\mathrm{dom}^\square(\Gamma_f)$ to $\mathrm{dom}^\square(\Gamma)$.

Note that, if $\Delta \vdash_c t : T$ then $\Gamma, \Delta \vdash t : T$. Hence, the well-typedness condition implies that $\gamma : \Gamma_f\varphi \rightsquigarrow \Gamma$ and thus that the left hand-side is well-typed: $\Gamma \vdash f\vec{l} : U\varphi\gamma$.

The positivity condition on the output type of f w.r.t. size variables appears in the previous works on sized types too. In [3], Abel gives an example of a function which is not terminating because it does not satisfy such a condition. This can be extended to more general continuity conditions [28, 1] and is indeed necessary (see Example 8).

As for the safeness condition, it simply says that one cannot do matching or have non-linearities on predicate variables, which is known to lead to non-termination [27]. It is also part of other works on the Calculus of Constructions with inductive types [36] and rewriting [40].

The positivity, safeness and accessibility conditions are decidable. For the conditions based on the computability closure, we prove the strong normalization in Section 7.

Let us now see some examples.

Example 3 (Division on natural numbers, Figure 1) Take the types $nat : \star$, $0 : nat^0$, $s : nat^\alpha \Rightarrow nat^{s\alpha}$, $- : nat^\alpha \Rightarrow nat^\beta \Rightarrow nat^\alpha$ and $/ : nat^\alpha \Rightarrow nat^\beta \Rightarrow$

[11] Primitive types are finitely branching.

[12] This implies in particular that every $x\Gamma$ is of the form $C^\epsilon\vec{t}$ with $\epsilon \in \mathcal{Z}$.

Figure 5: Computability closure of $f\vec{l} \to r$ with $f : (\vec{x} : \vec{T})U$ and $\gamma = \{\vec{x} \mapsto \vec{l}\}$

$$\text{(ax)} \quad \frac{}{\vdash_{\mathrm{c}} \star : \Box}$$

$$\text{(size)} \quad \frac{\vdash_{\mathrm{c}} \tau_C : \Box}{\vdash_{\mathrm{c}} C^a : \tau_C} \qquad (C \in \mathcal{CF}^{\Box})$$

$$\text{(symb)} \quad \frac{\vdash_{\mathrm{c}} \tau_g : s_g \quad (\forall i)\Delta \vdash_{\mathrm{c}} y_i\delta : U_i\psi\delta}{\Delta \vdash_{\mathrm{c}} g\vec{y}\delta : V\psi\delta} \qquad \begin{array}{l}(g \notin \mathcal{CF}^{\Box},\ g : (\vec{y} : \vec{U})V,\\ (g,\psi) <^{\mathcal{A}} (f,\varphi))\end{array}$$

$$\text{(var)} \quad \frac{\Delta \vdash_{\mathrm{c}} T : s_x}{\Delta, x : T \vdash_{\mathrm{c}} x : T} \qquad (x \notin \mathrm{dom}(\Delta))$$

$$\text{(weak)} \quad \frac{\Delta \vdash_{\mathrm{c}} t : T \quad \Delta \vdash_{\mathrm{c}} U : s_x}{\Delta, x : U \vdash_{\mathrm{c}} t : T} \qquad (x \notin \mathrm{dom}(\Delta))$$

$$\text{(prod)} \quad \frac{\Delta, x : U \vdash_{\mathrm{c}} V : s}{\Delta \vdash_{\mathrm{c}} (x : U)V : s}$$

$$\text{(abs)} \quad \frac{\Delta, x : U \vdash_{\mathrm{c}} v : V \quad \Delta \vdash_{\mathrm{c}} (x : U)V : s}{\Delta \vdash_{\mathrm{c}} [x : U]v : (x : U)V}$$

$$\text{(app)} \quad \frac{\Delta \vdash_{\mathrm{c}} t : (x : U)V \quad \Delta \vdash_{\mathrm{c}} u : U}{\Delta \vdash_{\mathrm{c}} tu : V\{x \mapsto u\}}$$

$$\text{(conv)} \quad \frac{\Delta \vdash_{\mathrm{c}} t : T \quad \Delta \vdash_{\mathrm{c}} T : s \quad \Delta \vdash_{\mathrm{c}} T' : s}{\Delta \vdash_{\mathrm{c}} t : T'} \qquad (T \leq T')$$

nat^{α}, with $\mathrm{Acc}(s) = \{1\}$. All positivity conditions are clearly satisfied. Safeness is immediate (there is no predicate variables). For the other conditions, we only detail (3) and (5).

- For (3), take $\Gamma_{-} = p : nat^{\alpha}, q : nat^{\beta}$, $\zeta_{-}(\alpha,\beta) = \alpha$, $\Gamma = x : nat^{\delta}, y : nat^{\epsilon}$, $\gamma = \{p \mapsto sx, q \mapsto sy\}$, $\varphi = \{\alpha \mapsto s\delta, \beta \mapsto s\epsilon\}$ and $s <_{\mathcal{F}} -$.
  - Well-typedness: By (symb), $\vdash_{\mathrm{c}} x : nat^{\delta}$ and $\vdash_{\mathrm{c}} y : nat^{\epsilon}$. Thus, by (symb), $\vdash_{\mathrm{c}} sx : nat^{s\delta}$ and $\vdash_{\mathrm{c}} sy : nat^{s\epsilon}$.
  - Accessibility: One can easily check that $sx : nat^{s\delta} \gg_{\varphi} x : nat^{\delta}$ and $sy^{s\epsilon} \gg_{\varphi} y : nat^{\epsilon}$.
  - Computability closure: By (symb), $\vdash_{\mathrm{c}} x : nat^{\delta}$ and $\vdash_{\mathrm{c}} y : nat^{\epsilon}$. By (symb), $\vdash_{\mathrm{c}} -xy : nat^{\delta}$ since $\zeta_{-}(\delta,\epsilon) = \delta < \zeta_{-}(s\delta, s\epsilon) = s\delta$. Thus, by (sub), $\vdash_{\mathrm{c}} -xy : nat^{s\delta}$.
- For (5), take $\Gamma_{/} = p : nat^{\alpha}, q : nat^{\beta}$, $\zeta_{/}(\alpha,\beta) = \alpha$, $\Gamma = x : nat^{\delta}, y : nat^{\epsilon}$,

$\gamma = \{p \mapsto sx, q \mapsto y\}$, $\varphi = \{\alpha \mapsto s\delta, \beta \mapsto \epsilon\}$ and $- <_{\mathcal{F}} /$.

- Well-typedness: By (symb), $\vdash_{\mathrm{c}} x : nat^{\delta}$ and $\vdash_{\mathrm{c}} y : nat^{\epsilon}$. Thus, by (symb), $\vdash_{\mathrm{c}} sx : nat^{s\delta}$.
- Accessibility: One can easily check that $sx : nat^{s\delta} \gg_{\varphi} x : nat^{\delta}$ and $y : nat^{\epsilon} \gg_{\varphi} y : nat^{\epsilon}$.
- Computability closure: By (symb), $\vdash_{\mathrm{c}} x : nat^{\delta}$ and $\vdash_{\mathrm{c}} y : nat^{\epsilon}$. By (symb), $\vdash_{\mathrm{c}} -xy : nat^{\delta}$. By (symb), $\vdash_{\mathrm{c}} /(-xy)y : nat^{\delta}$ since $\zeta_{/}(\delta, \epsilon) = \delta < \zeta_{/}(s\delta, \epsilon) = s\delta$. Thus, by (symb), $\vdash_{\mathrm{c}} s(/(-xy)y) : nat^{s\delta}$.

Example 4 (Addition on Brouwer's ordinals, Figure 2) Take the types $ord : \star$, $0 : nat^{0}$, $s : nat^{\alpha} \Rightarrow nat^{s\alpha}$, $lim : (nat \Rightarrow ord^{\alpha}) \Rightarrow ord^{s\alpha}$ and $+ : nat^{\alpha} \Rightarrow nat^{\beta} \Rightarrow nat^{\infty}$, with $\mathrm{Acc}(s) = \mathrm{Acc}(lim) = \{1\}$. All positivity conditions are clearly satisfied. We only detail rule (3). Take $\Gamma_{+} = p : ord^{\alpha}, q : ord^{\beta}$, $\zeta_{+}(\alpha, \beta) = \alpha$, $\Gamma = f : nat^{\infty} \Rightarrow ord^{\delta}, y : ord^{\epsilon}$, $\gamma = \{p \mapsto limf, q \mapsto y\}$, $\varphi = \{\alpha \mapsto s\delta, \beta \mapsto \epsilon\}$ and $s, lim <_{\mathcal{F}} +$.

- Well-typedness: By (symb), $\vdash_{\mathrm{c}} f : nat^{\infty} \Rightarrow ord^{\delta}$ and $\vdash_{\mathrm{c}} y : ord^{\epsilon}$. Thus, by (symb), $\vdash_{\mathrm{c}} limf : ord^{s\delta}$.
- Accessibility: One can easily check that $limf : ord^{s\delta} \gg_{\varphi} f : nat^{\infty} \Rightarrow ord^{\delta}$ and $y : ord^{\epsilon} \gg_{\varphi} y : ord^{\epsilon}$.
- Computability closure: By (symb), $\vdash_{\mathrm{c}} f : nat^{\infty} \Rightarrow ord^{\delta}$ and $\vdash_{\mathrm{c}} y : ord^{\epsilon}$. Let $\Delta = x : nat^{\infty}$. By (var), $\Delta \vdash_{\mathrm{c}} x : nat^{\infty}$. By (weak), $\Delta \vdash_{\mathrm{c}} f : nat^{\infty} \Rightarrow ord^{\delta}$ and $\Delta \vdash_{\mathrm{c}} y : ord^{\epsilon}$. By (app), $\Delta \vdash_{\mathrm{c}} fx : ord^{\delta}$. By (symb), $\Delta \vdash_{\mathrm{c}} +(fx)y : ord^{\infty}$ since $\zeta_{+}(\delta, \epsilon) = \delta < \zeta_{+}(s\delta, \epsilon) = s\delta$. By (abs), $\vdash_{\mathrm{c}} [x : nat^{\infty}](+(fx)y) : (x : nat^{\infty})ord^{\delta}$. Thus, by (symb), $\vdash_{\mathrm{c}} lim([x : nat^{\infty}](+(fx)y)) : ord^{s\delta}$.

Example 5 (Quick sort, Figure 6) Take the types $bool : \star$, $true : bool^{\infty}$, $false : bool^{\infty}$, $list : \star$, $nil : list^{0}$, $cons : nat^{\infty} \Rightarrow list^{\alpha} \Rightarrow list^{s\alpha}$, $blist : \star$, $pair : list^{\alpha} \Rightarrow list^{\beta} \Rightarrow blist^{max(\alpha,\beta)}$, $fst : blist^{\alpha} \Rightarrow list^{\alpha}$, $snd : blist^{\alpha} \Rightarrow list^{\alpha}$, $\leq : nat^{\infty} \Rightarrow nat^{\infty} \Rightarrow bool^{\infty}$, $pivot : nat^{\infty} \Rightarrow list^{\alpha} \Rightarrow blist^{\alpha}$, $qs : list^{\infty} \Rightarrow list^{\infty} \Rightarrow list^{\infty}$ and $qsort : list^{\infty} \Rightarrow list^{\infty}$. We only detail the computability closure condition of rule (11).

Take $\zeta_{qs}(\alpha, \beta) = \alpha$, $\Gamma = x : nat^{\infty}, l : list^{\delta}, l' : list^{\epsilon}$, $\varphi = \{\alpha \mapsto s\delta, \beta \mapsto \epsilon\}$ and $qs >_{\mathcal{F}} pivot >_{\mathcal{F}} cons, pair, fst, snd$. By (symb), $\vdash_{\mathrm{c}} x : nat^{\infty}$, $\vdash_{\mathrm{c}} l : list^{\delta}$ and $\vdash_{\mathrm{c}} l' : list^{\epsilon}$. By (symb), $\vdash_{\mathrm{c}} pivot\ x\ l : blist^{\delta}$. By (symb), $\vdash_{\mathrm{c}} u : list^{\delta}$ and $\vdash_{\mathrm{c}} v : list^{\delta}$. By (symb), $\vdash_{\mathrm{c}} qs\ v\ l' : list^{\infty}$. By (symb), $\vdash_{\mathrm{c}} cons\ x\ (qs\ v\ l') : list^{\infty}$. Thus, by (symb), $\vdash_{\mathrm{c}} qs\ u\ (cons\ x\ (qs\ v\ l')) : list^{\infty}$ since $\zeta_{qs}(\delta, \infty) = \delta < \zeta_{qs}(s\delta, \epsilon) = s\delta$.

Note that we cannot take $qs : list^{\alpha} \Rightarrow list^{\beta} \Rightarrow list^{\alpha+\beta}$ and thus $qsort : list^{\alpha} \Rightarrow list^{\alpha}$ since too much information is lost by taking $pair : list^{\alpha} \Rightarrow list^{\beta} \Rightarrow blist^{max(\alpha,\beta)}$. Even though we take $pair : list^{\alpha} \Rightarrow list^{\beta} \Rightarrow blist^{\langle\alpha,\beta\rangle}$ with $\langle\alpha, \beta\rangle$ interpreted as a pair of ordinals, the current setting does not allow us to say that $pivot$ has type $nat^{\infty} \Rightarrow list^{\alpha} \Rightarrow blist^{\langle\beta,\gamma\rangle}$ for some β and γ such that $\beta + \gamma = \alpha$, as it can be done in Xi's framework [42].

The following examples are taken from [25].

Figure 6: Quick sort

$$
\begin{array}{lrcl}
(1) & \mathit{fst}\ (\mathit{pair}\ x\ y) & \to & x \\
(2) & \mathit{snd}\ (\mathit{pair}\ x\ y) & \to & y \\
(3) & \leq\ 0\ x & \to & \mathit{true} \\
(4) & \leq\ (s\ x)\ 0 & \to & \mathit{false} \\
(5) & \leq\ (s\ x)\ (s\ y) & \to & \leq\ x\ y \\
(6) & \mathit{if}\ \mathit{true}\ x\ y & \to & x \\
(7) & \mathit{if}\ \mathit{false}\ x\ y & \to & y \\
(8) & \mathit{pivot}\ x\ \mathit{nil} & \to & \mathit{pair}\ \mathit{nil}\ \mathit{nil} \\
(9) & \mathit{pivot}\ x\ (\mathit{cons}\ y\ l) & \to & \mathit{if}\ (\leq\ y\ x)\ (\mathit{pair}\ (\mathit{cons}\ y\ u)\ v)\ (\mathit{pair}\ u\ (\mathit{cons}\ y\ v)) \\
 & & & \text{where } u = \mathit{fst}\ (\mathit{pivot}\ x\ l) \text{ and } v = \mathit{snd}\ (\mathit{pivot}\ x\ l) \\
(10) & \mathit{qs}\ \mathit{nil}\ l & \to & l \\
(11) & \mathit{qs}\ (\mathit{cons}\ x\ l)\ l' & \to & \mathit{qs}\ u\ (\mathit{cons}\ x\ (\mathit{qs}\ v\ l')) \\
 & & & \text{where } u = \mathit{fst}\ (\mathit{pivot}\ x\ l) \text{ and } v = \mathit{snd}\ (\mathit{pivot}\ x\ l) \\
(12) & \mathit{qsort}\ l & \to & \mathit{qs}\ l\ \mathit{nil}
\end{array}
$$

Figure 7: Paulson's normalization of *if*-expressions

$$
\begin{array}{lrcl}
(1) & \mathit{nm}\ \mathit{at} & \to & \mathit{at} \\
(2) & \mathit{nm}\ (\mathit{if}\ \mathit{at}\ y\ z) & \to & \mathit{if}\ \mathit{at}\ (\mathit{nm}\ y)\ (\mathit{nm}\ z) \\
(3) & \mathit{nm}\ (\mathit{if}\ (\mathit{if}\ u\ v\ w)\ y\ z) & \to & \mathit{nm}\ (\mathit{if}\ u\ (\mathit{nm}\ (\mathit{if}\ v\ y\ z))\ (\mathit{nm}\ (\mathit{if}\ w\ y\ z)))
\end{array}
$$

Example 6 (Paulson's normalization of *if*-expressions, Figure 7) Take the types $\mathit{expr} : \star$, $\mathit{at} : \mathit{expr}^1$, $\mathit{if} : \mathit{expr}^\alpha \Rightarrow \mathit{expr}^\beta \Rightarrow \mathit{expr}^\gamma \Rightarrow \mathit{expr}^{\alpha(1+\beta+\gamma)}$ and $\mathit{nm} : \mathit{expr}^\alpha \Rightarrow \mathit{expr}^\alpha$. We only detail the computability closure condition of rule (3). Take $\zeta_{nm}(\alpha) = \alpha$, $\Gamma = u : \mathit{expr}^\alpha, v : \mathit{expr}^\beta, w : \mathit{expr}^\gamma, y : \mathit{expr}^\delta, z : \mathit{expr}^\epsilon$, $\upsilon = \alpha(1+\beta+\gamma)(1+\delta+\epsilon)$, $\varphi = \{\alpha \mapsto \upsilon\}$ and $\mathit{nm} >_\mathcal{F} \mathit{at}, \mathit{if}$. Then, one can check that υ is strictly greater than $\beta(1+\delta+\epsilon)$, $\gamma(1+\delta+\epsilon)$ and $\alpha(1+\beta(1+\delta+\epsilon)+\gamma(1+\delta+\epsilon))$ if variables are interpreted by strictly positive integers.

Example 7 (Huet and Hullot's reverse function, Figure 8) Take the types $\mathit{rev}1 : \mathit{nat}^\infty \Rightarrow \mathit{list}^\infty \Rightarrow \mathit{nat}^\infty$, $\mathit{rev}2 : \mathit{nat}^\infty \Rightarrow \mathit{list}^\beta \Rightarrow \mathit{list}^\beta$ and $\mathit{rev} : \mathit{list}^\alpha \Rightarrow \mathit{list}^\alpha$. We only detail the computability closure condition of rule (4). Take $\zeta_{rev}(\alpha) = 2\alpha$, $\zeta_{rev2}(\alpha, \beta) = 2\beta + 1$, $\Gamma = x : \mathit{nat}^\infty, y : \mathit{nat}^\infty, l : \mathit{list}^\delta$, $\varphi = \{\beta \mapsto \delta + 1\}$ and $\mathit{rev} \simeq_\mathcal{F} \mathit{rev}2 >_\mathcal{F} \mathit{rev}1 >_\mathcal{F} \mathit{cons}, \mathit{nil}$. Then, one can check that $\zeta_{rev2}(\infty, \delta+1) = 2\delta+3$ is strictly greater than $\zeta_{rev2}(\infty, \delta) = 2\delta+1$,

Figure 8: Huet and Hullot's reverse function

$$
\begin{array}{lrcl}
(1) & rev1\ x\ nil & \to & x \\
(2) & rev1\ x\ (cons\ y\ l) & \to & rev1\ y\ l \\
(3) & rev2\ x\ nil & \to & nil \\
(4) & rev2\ x\ (cons\ y\ l) & \to & rev\ (cons\ x\ (rev\ (rev2\ y\ l))) \\
(5) & rev\ nil & \to & nil \\
(6) & rev\ (cons\ x\ l) & \to & cons\ (rev1\ x\ l)\ (rev2\ x\ l)
\end{array}
$$

$\zeta_{rev}(\delta) = 2\delta$ and $\zeta_{rev}(1+\delta) = 2\delta + 2$.

Figure 9: Mac Carthy's "91" function

$$
\begin{array}{lrcll}
(1) & f\ x & \to & f\ (f\ (+\ x\ 11)) & \text{if } \leq\ x\ 100 = true \\
(2) & f\ x & \to & -\ x\ 10 & \text{if } \leq\ x\ 100 = false
\end{array}
$$

Example 8 (Mac Carthy's "91" function, Figure 9) Mac Carthy's "91" function f is defined by the following equations: $f(x) = f(f(x+11))$ if $x \leq 100$, and $f(x) = x - 10$ otherwise. In fact, one can prove that f is equal to the function F such that $F(x) = 91$ if $x \leq 100$, and $F(x) = x - 10$ otherwise. A way to formalize this in CACSA would be to use conditional rewrite rules (see Figure 9) and take[13] $f : nat^{\alpha} \Rightarrow nat^{F(\alpha)}$ and $\zeta_f^X(x) = max(0, 101 - x)$ as measure function, as it can be done in Xi's framework. Then, by taking into account the rewrite rule conditions, one could prove that, if $\Gamma = x : nat^{\delta}$ and $\leq\ x\ 100 = true$, then $\delta \leq 100$, $\zeta_f(\delta + 11) < \zeta_f(\delta)$ and $\zeta_f(F(\delta)) < \zeta_f(\delta)$.

7 Termination proof

We first prove some lemmas for proving the correctness of accessibility w.r.t. computability (accessible subterms of a computable term are computable). Then, we prove the correctness of the computability closure (every term of the computability closure is computable) and the computability of every symbol, hence the strong normalization of every well-typed term.

Lemma 36 (Accessibility properties)
(1) If $t : T \rhd^k u : D^e \vec{u}$ then $T = C^{s^k e} \vec{t}$.
(2) If $t : C^{\beta} \vec{t} \gg_{\varphi} u : U$ then there are $\epsilon \in \mathcal{Z}$ and $k \geq 0$ such that $\beta\varphi = s^k \epsilon$.

[13] Note that $F(\alpha)$ is monotone w.r.t. α.

(3) If $t : T \rhd u : U$, $t\sigma \in K^{\mathfrak{b}}_C(\vec{S})$ then $o_{C(\vec{S})}(t)$ is a successor ordinal.

(4) If $t : T \rhd u : U$ and $t\sigma \in I^{\mathfrak{b}}_C(\vec{S})$ then $u\sigma \in I^{\mathfrak{b}}_D(\vec{S'})$ for some D and $\vec{S'}$.

(5) Let $f : (\vec{y} : \vec{U})C^{s\alpha}\vec{v}$ be a finitely branching constructor such that, if $j \in \mathrm{Acc}(f)$ and $\mathrm{Pos}(\alpha, U_j) \neq \emptyset$ then $U_j = C^{\alpha}_j\vec{v^j}$. If $f\vec{u} \in K^{\mathfrak{a}}_C(\vec{S})$ then $o_{C(\vec{S})}(f\vec{u}) = max\{o_{C_j(\vec{S^j})}(u_j) \mid j \in \mathrm{Acc}(f), \mathrm{Pos}(\alpha, U_j) \neq \emptyset\} + 1$, where $\vec{S^j} = [\![\vec{v^j}]\!]^{\nu}_{\xi,\theta}$, $y\xi = S_{\iota_y}$, $\vec{y}\theta = \vec{u}$ and $\alpha\nu = \mathfrak{a}$.

(6) If $t : T \rhd^k \rhd\ u : U$ and $t\sigma \in K^{\mathfrak{b}}_C(\vec{S})$ then $o_{C(\vec{S})}(t) = \mathfrak{a} + k + 1$ for some $\mathfrak{a}$.

(7) If $t : T \rhd^* u : U$ and $t\sigma \in [\![T]\!]^{\mu}_{\xi,\sigma}$ then $u\sigma \in [\![U]\!]^{\mu}_{\xi,\sigma}$.

Proof.

(1) By induction on k. For $k = 0$, this is immediate. Assume now that $t : T \rhd^k v : V \rhd_a u : D^e\vec{u}$. Then, $a = e$ and $V = E^{se}\vec{v}\gamma$. Therefore, by induction hypothesis, $T = C^{s^{k+1}e}\vec{t}$.

(2) There are two cases.
- $t : C^{\beta}\varphi = u : U$ and $\varphi_{|\mathcal{V}(T)}$ is a renaming. Take $\epsilon = \beta\varphi$ and $k = 0$.
- $t : C^{\beta}\varphi \rhd^k v : V \rhd_{\epsilon} u : U$. Then, $V = D^{s\epsilon}\vec{v}$ and, by (1), $\beta\varphi = s^{k+1}\epsilon$.

(3) By Lemma 30.

(4) By (3), we can assume that $t\sigma \in I^{\mathfrak{a}+1}_C(\vec{S})$. By Definition 29, $u_j \in [\![U_j]\!]^{\nu}_{\xi,\theta}$ with $y\xi = S_{\iota_y}$, $\vec{y}\theta = \vec{u}$ and $\alpha\nu = \mathfrak{a}$. By definition of $\rhd$, $U_j = D^{\alpha}\vec{u}$. Thus, $u_j \in I^{\mathfrak{a}}_D(\vec{S'})$ with $\vec{S'} = [\![\vec{u}]\!]^{\nu}_{\xi,\theta}$.

(5) By (3), we can assume that $f\vec{u} \in I^{\mathfrak{a}+1}_C(\vec{S})$. By (4), for all $j \in \mathrm{Acc}(f)$ such that $\mathrm{Pos}(\alpha, U_j) \neq \emptyset$, $u_j \in I^{\mathfrak{a}}_{C_j}(\vec{S^j})$. Let $\mathfrak{a}_j = o_{C_j(\vec{S^j})}(u_j)$. Since $\mathfrak{a}$ is as small as possible, we must have $max\{\mathfrak{a}_j \mid j \in \mathrm{Acc}(f), \mathrm{Pos}(\alpha, U_j) \neq \emptyset\} = \mathfrak{a}$.

(6) By induction on k. For $k = 0$, this is (3). Assume now that $t : T \rhd u : U \rhd^k \rhd\ v : V$. By (4), for all $j \in \mathrm{Acc}(f)$, $u_j\sigma \in I^{\mathfrak{a}}_{D_j}(\vec{S^j})$. Let $\mathfrak{a}_j = o_{C_j(\vec{S^j})}(u_j\sigma)$. By induction hypothesis, $\mathfrak{a}_j = \mathfrak{b}_j + k + 1$. Therefore, by (5), $o_{C(\vec{S})}(t\sigma) = \mathfrak{b}_j + k + 2$ for some $\mathfrak{b}_j$.

(7) By induction on the number of $\rhd$-steps. If there is no step, this is immediate. Assume now that $t : T \rhd_a u : U \rhd^* v : V$ and $\alpha\varphi = a$. Since $T = C^{s\alpha\varphi}\vec{v}\gamma$, $[\![T]\!]^{\mu}_{\xi,\sigma} = I^{\alpha\varphi\mu+1}_C(\vec{S})$ with $\vec{S} = [\![\vec{v}\gamma]\!]^{\mu}_{\xi,\sigma}$. Therefore, $u\sigma \in [\![U_j]\!]^{\varphi\mu}_{\eta,\gamma\sigma}$ with $y\eta = S_{\iota_y}$. Since $v_{\iota_y} = y$, $y\eta = [\![y\gamma]\!]^{\varphi\mu}_{\xi,\sigma} = [\![y\gamma]\!]^{\mu}_{\xi,\sigma}$ since $\mathrm{Pos}(\alpha, \gamma) = \emptyset$. So, by candidate substitution, $[\![U_j]\!]^{\varphi\mu}_{\eta,\gamma\sigma} = [\![U_j\gamma]\!]^{\varphi\mu}_{\xi,\sigma} = [\![U]\!]^{\mu}_{\xi,\sigma}$. Therefore, by induction hypothesis, $v\sigma \in [\![V]\!]^{\mu}_{\xi,\sigma}$. ■

Theorem 37 (Accessibility correctness) If $t : T \gg_{\varphi} u : U$, $T = C^{\beta}\vec{t}$, $\mathcal{V}(\vec{t}) = \emptyset$ and $t\sigma \in [\![T]\!]^{\mu}_{\xi,\sigma}$ then there exists ν such that $\beta\varphi\nu \leq \beta\mu$ and $u\sigma \in [\![U]\!]^{\nu}_{\xi,\sigma}$.

Proof. There are two cases:

- $t : T\varphi = u : U$ and $\varphi_{|\mathcal{V}(T)}$ is a renaming. Let $\nu = \varphi^{-1}_{|\mathcal{V}(T)}\mu$. $\beta\varphi\nu = \beta\mu$ and $u\sigma = t\sigma \in [\![T]\!]^{\mu}_{\xi,\sigma} = [\![T\varphi]\!]^{\nu}_{\xi,\sigma}$.

- $t : T\varphi \rhd^* u : U \rhd_\epsilon v : V$. By definition of $\rhd_\epsilon$, $U = D^{s\epsilon}\vec{u}$. By Lemma 36 (1), $\beta\varphi = s^{k+1}\epsilon$. By (6), there exists $\mathfrak{a}$ such that $\mathfrak{a} + k + 1 \leq \beta\mu$ and $t\sigma \in I_C^{\mathfrak{a}+k+1}(\vec{S})$. Let $\epsilon\nu = \mathfrak{a}$. Then, $\beta\varphi\nu = s^{k+1}\epsilon\nu = \mathfrak{a} + k + 1 \leq \beta\mu$, $t\sigma \in [\![T\varphi]\!]^\nu_{\xi,\sigma}$ and, by (7), $u\sigma \in [\![T\varphi]\!]^\nu_{\xi,\sigma}$. ■

Theorem 38 (Correctness of the computability closure) Let $(f\vec{l} \to r, \Gamma, \varphi) \in \mathcal{R}$, $f : (\vec{x} : \vec{T})U$ and $\gamma = \{\vec{x} \mapsto \vec{l}\}$. Assume that, for all $(g, \mu) <^{\mathfrak{A}} (f, \varphi\nu)$, $g \in [\![\tau_g]\!]^\mu$. If $\Delta \vdash_{\mathrm{c}} t : T$ and $\xi, \sigma \models_\nu \Gamma, \Delta$ then $t\sigma \in [\![T]\!]^\nu_{\xi,\sigma}$.

Proof. By induction on $\Delta \vdash_{\mathrm{c}} t : T$. We only detail the case (symb). Since $(g, \psi) <^{\mathcal{A}} (f, \varphi)$, $(g, \psi\nu) <^{\mathfrak{A}} (f, \varphi\nu)$. Hence, by assumption, $g \in [\![\tau_g]\!]^{\psi\nu}$. Now, by induction hypothesis, $\vec{y}\delta\sigma \in [\![\vec{U}\psi\delta]\!]^\nu_{\xi,\sigma}$. By candidate substitution, there exists η such that $[\![\vec{U}\psi\delta]\!]^\nu_{\xi,\sigma} = [\![\vec{U}\psi]\!]^\nu_{\eta,\delta\sigma}$. By size substitution, $[\![\vec{U}\psi]\!]^\nu_{\eta,\delta\sigma} = [\![\vec{U}]\!]^{\psi\nu}_{\eta,\delta\sigma}$. Therefore, $g\vec{y}\delta\sigma \in [\![V]\!]^{\psi\nu}_{\eta,\delta\sigma} = [\![V\psi\delta]\!]^\nu_{\xi,\sigma}$.

Lemma 39 (Computability of symbols) For all f and μ, $f \in [\![\tau_f]\!]^\mu$.

Proof. Assume that $\tau_f = (\vec{x} : \vec{T})U$ with U distinct from a product. $f \in [\![\tau_f]\!]^\mu$ iff, for all η, θ such that $\eta, \theta \models_\mu \Gamma_f$, $f\vec{x}\theta \in [\![U]\!]^\mu_{\eta,\theta}$. We prove it by induction on $((f, \mu), \theta)$ with $(>^{\mathfrak{A}}, \to)_{\mathrm{lex}}$ as well-founded ordering. Let $t_i = x_i\theta$ and $t = f\vec{t}$. By assumption, for every rule $f\vec{l} \to r \in \mathcal{R}$, $|\vec{l}| \leq |\vec{t}|$. So, if $f \notin Cons$ then t is neutral and it suffices to prove that $\to(t) \subseteq [\![U]\!]^\mu_{\eta,\theta}$. Otherwise, $[\![U]\!]^\mu_{\eta,\theta} = I_C^{a\mu}(\vec{S})$ with $\vec{S} = [\![\vec{v}]\!]^\mu_{\eta,\theta}$. Since $\eta, \theta \models_\mu \Gamma_f$, $t_j \in [\![T_j]\!]^\mu_{\eta,\theta}$. Therefore, in this case too, it suffices to prove that $\to(t) \subseteq [\![U]\!]^\mu_{\eta,\theta}$.

If the reduction takes place in one t_i then we can conclude by induction hypothesis. Assume now that there exist $(l \to r, \Gamma, \varphi) \in \mathcal{R}$ and σ such that $t = l\sigma$. Then, $l = f\vec{l}$ and $\theta = \gamma\sigma$ with $\gamma = \{\vec{x} \mapsto \vec{l}\}$.

We now define ξ such that $[\![U]\!]^\mu_{\eta,\gamma\sigma} = [\![U\gamma]\!]^\mu_{\xi,\sigma}$ and $[\![\vec{T}]\!]^\mu_{\eta,\gamma\sigma} = [\![\vec{T}\gamma]\!]^\mu_{\xi,\sigma}$. By safeness, γ is an injection from $\mathrm{dom}^\square(\Gamma_f)$ to $\mathrm{dom}^\square(\Gamma)$. Let $y \in \mathrm{dom}^\square(\Gamma)$. If there exists $x \in \mathrm{dom}(\Gamma_f)$ (necessarily unique) such that $y = x\gamma$, we take $y\xi = x\eta$. Otherwise, we take $y\xi = \top_{y\Gamma}$.

We check that $\xi \models \Gamma$. If $y \neq x\gamma$, $y\xi = \top_{y\Gamma} \in \mathcal{R}_{y\Gamma}$. If $y = x\gamma$ then $y\xi = x\eta$. Since $\eta \models \Gamma_f$, $x\eta \in \mathcal{R}_{x\Gamma_f}$. Since $\gamma : \Gamma_f\varphi \rightsquigarrow \Gamma$, $\Gamma \vdash y : x\Gamma_f\varphi\gamma$. Therefore, $y\Gamma \leq x\Gamma_f\varphi\gamma$ and $\mathcal{R}_{y\Gamma} = \mathcal{R}_{x\Gamma_f\varphi\gamma} = \mathcal{R}_{x\Gamma_f}$. So, $\xi \models \Gamma$.

Now, by candidate substitution, $[\![U\gamma]\!]^\mu_{\xi,\sigma} = [\![U]\!]^\mu_{\eta',\gamma\sigma}$ with $x\eta' = [\![x\gamma]\!]_{\xi,\sigma}$. Let $x \in \mathrm{FV}(\vec{T}U)$. By safeness, $x\gamma = y \in \mathrm{dom}^\square(\Gamma)$ and $x\eta' = y\xi = x\eta$. Therefore, $\eta' = \eta$.

We now prove that $\xi, \sigma \models_\nu \Gamma$ for some valuation ν such that $\varphi\nu \leq \mu$. Let $x \in \mathrm{dom}(\Gamma)$. By assumption, there exists i such that $l_i : T_i\gamma \gg_\varphi x : x\Gamma$, $T_i\gamma = C^{\beta_x}\vec{u}$ and $\mathcal{V}(\vec{u}) = \emptyset$. By Lemma 36 (2), there is ϵ_x and k_x such that $\beta_x\varphi = s^{k_x}\epsilon_x$. Since $l_i\sigma \in [\![T_i\gamma]\!]_{\xi,\sigma}$, by Theorem 37, there exists ν_x such that $x\sigma \in [\![x\Gamma]\!]^{\nu_x}_{\xi,\sigma}$ and $\beta_x\varphi\nu_x \leq \beta_x\mu$. Since Γ is linear w.r.t. size variables, $\epsilon_x \neq \epsilon_y$ whenever $x \neq y$. So, we can define ν by taking $\epsilon_x\nu = \epsilon_x\nu_x$. Then, $\beta_x\varphi\nu = s^{k_x}\epsilon_x\nu = s^{k_x}\epsilon_x\nu_x = \beta_x\varphi\nu_x \leq \beta_x\mu$.

Therefore, since $\vdash_c r : U\varphi\gamma$, by correctness of the computability closure, $r\sigma \in [\![U\varphi\gamma]\!]^{\nu}_{\xi,\sigma} = [\![U\varphi]\!]^{\nu}_{\eta,\theta} = [\![U]\!]^{\varphi\nu}_{\eta,\theta} \leq [\![U]\!]^{\mu}_{\eta,\theta}$ since, for all α, $\mathrm{Pos}(\alpha, U) \subseteq \mathrm{Pos}^{+}(U)$. ■

Theorem 40 (Strong normalization) Every well-typed term is strongly normalizable.

Proof. The invariance by reduction is proved in [11]. Hence, we can conclude by Theorem 25 and Lemma 39. ■

8 Conclusion

The notion of computability closure, first introduced in [12] and further extended to higher-order pattern-matching [10], higher-order recursive path ordering [29], type-level rewriting [7] and rewriting modulo equational theories [9], again shows to be essential for extending to rewriting and dependent types type-based termination criteria for (polymorphic) λ-calculi with inductive types and case analysis [28, 42, 5, 2]. In contrast with what is suggested in [5], this notion, which is expressed as a sub-system of the whole type system (by restricting the size of arguments in function calls in some computability-preserving way, see Figure 5), allows pattern-matching and does not suffer from limitations one could find in systems relying on external guard predicates for recursive definitions.

Moreover, we allow a richer size algebra than the one in [28, 5, 2] (see Section 6). But, we do not allow existential size variables and conditional rewriting that are essential for capturing for instance the size-preserving property of quicksort (Example 5) and Mac Carty's "91" function (Example 8) respectively, as it can be done in Xi's work [42]. Such extensions should allow us to subsume Xi's work completely. More generally, it is important to have a better understanding of the differences between Xi's work which does not use subtyping (but has existential size variables and singleton types) and the other works that are based on subtyping.

In this work, we assume that users provide appropriate sized types for function symbols and then check by our technique that the rewrite rules defining these function symbols are compatible with their types. An important extension would be to infer these types. Works in this direction for ML-like languages are [32, 43, 17]. The exact relations between these works and with refinement types also [33, 22] still have to be investigated. Note also that deciding the non-size-increasing property of some functions is investigated in [23, 24].

We made two important assumptions that also need further research. First, the confluence of $\beta \cup \mathcal{R}$, which is still an open problem when $\mathcal{R}$ is confluent, terminating, non left-linear and contains type-level rewrite rules. Second, the preservation of typing under rewriting (subject reduction for $\mathcal{R}$), for which we need to find decidable sufficient conditions (see Example 1).

Finally, by combining rewriting and subtyping in the Calculus of Constructions, this work may also be seen as an important step towards the integration of membership equational logic [13] and dependent type systems. Previous works

in this direction are [6, 14, 37].

Acknowledgments. I would like to thank very much Ralph Matthes for having invited me for a one-week stay in Münich in February 2002. Andreas Abel's technical report [2] and the discussions I had with Ralph and Andreas about monotone inductive types and termination were the starting point of the present work.

References

[1] A. Abel. Termination and productivity checking with continuous types. In *Proceedings of the 6th International Conference on Typed Lambda Calculi and Applications*, Lecture Notes in Computer Science 2701, 2003.

[2] A. Abel. Termination checking with types. Technical Report 0201, Ludwig Maximilians Universität, München, Germany, 2002.

[3] A. Abel. Termination checking with types, 2003. Submitted to ITA.

[4] H. Barendregt. Lambda calculi with types. In S. Abramski, D. Gabbay, and T. Maibaum, editors, *Handbook of logic in computer science*, volume 2. Oxford University Press, 1992.

[5] G. Barthe, M. J. Frade, E. Giménez, L. Pinto, and T. Uustalu. Type-based termination of recursive definitions. *Mathematical Structures in Computer Science*, 14(1):97–141, 2004.

[6] G. Barthe and F. van Raamsdonk. Constructor subtyping in the calculus of inductive constructions. In *Proceedings of the 3rd International Conference on Foundations of Software Science and Computation Structures*, Lecture Notes in Computer Science 1784, 2000.

[7] F. Blanqui. Definitions by rewriting in the Calculus of Constructions (extended abstract). In *Proceedings of the 16th IEEE Symposium on Logic in Computer Science*, 2001.

[8] F. Blanqui. Inductive types in the Calculus of Algebraic Constructions. In *Proceedings of the 6th International Conference on Typed Lambda Calculi and Applications*, Lecture Notes in Computer Science 2701, 2003.

[9] F. Blanqui. Rewriting modulo in Deduction modulo. In *Proceedings of the 14th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science 2706, 2003.

[10] F. Blanqui. Termination and confluence of higher-order rewrite systems. In *Proceedings of the 11th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science 1833, 2000.

[11] F. Blanqui. Definitions by rewriting in the Calculus of Constructions, 2003. To appear in Mathematical Structures in Computer Science.

[12] F. Blanqui, J.-P. Jouannaud, and M. Okada. Inductive-data-type Systems. *Theoretical Computer Science*, 272:41–68, 2002.

[13] A. Bouhoula, J.-P. Jouannaud, and J. Meseguer. Specification and proof in membership equational logic. *Theoretical Computer Science*, 236:35–132, 2000.

[14] G. Castagna and G. Chen. Dependent types with subtyping and late-bound overloading. *Information and Computation*, 168(1):1–67, 2001.

[15] G. Chen. *Subtyping, Type Conversion and Transitivity Elimination.* PhD thesis, Université Paris VII, France, 1998.

[16] W. N. Chin and M. Hagiya. A bounds inference method for vector-based memoisation. In *Proceedings of the 2nd ACM International Conference on Functional Programming*, SIGPLAN Notices 32(8), 1997.

[17] W. N. Chin and S. C. Khoo. Calculating sized types. *Journal of Higher-Order and Symbolic Computation*, 14(2–3):261–300, 2001.

[18] T. Coquand. An analysis of Girard's paradox. In *Proceedings of the 1st IEEE Symposium on Logic in Computer Science*, 1986.

[19] T. Coquand and G. Huet. The Calculus of Constructions. *Information and Computation*, 76(2–3):95–120, 1988.

[20] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 6. North-Holland, 1990.

[21] G. Dowek and B. Werner. Proof normalization modulo. In *Proceedings of the International Workshop on Types for Proofs and Programs*, Lecture Notes in Computer Science 1657, 1998.

[22] T. Freeman. *Refinement types for ML.* PhD thesis, Carnegie Mellon University, United States, 1994.

[23] J. Giesl. Automated termination proofs with measure functions. In *Proceedings of the 19thGerman Conference on Artificial Intelligence1995*, Lecture Notes in Computer Science 981.

[24] J. Giesl. Termination analysis for functional programs using term orderings. In *Proceedings of the 2nd International Symposium on Static Analysis*, Lecture Notes in Computer Science 983, 1995.

[25] J. Giesl. Termination of nested and mutually recursive algorithms. *Journal of Automated Reasoning*, 19(1):1–29, 1997.

[26] E. Giménez. Structural recursive definitions in type theory. In *Proceedings of the 25th International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science 1443, 1998.

[27] R. Harper and J. Mitchell. Parametricity and variants of Girard's J operator. *Information Processing Letters*, 70:1–5, 1999.

[28] J. Hughes, L. Pareto, and A. Sabry. Proving the correctness of reactive systems using sized types. In *Proceedings of the 23th ACM Symposium on Principles of Programming Languages, 1996.*

[29] J.-P. Jouannaud and A. Rubio. The Higher-Order Recursive Path Ordering. In *Proceedings of the 14th IEEE Symposium on Logic in Computer Science*, 1999.

[30] J. W. Klop, V. van Oostrom, and F. van Raamsdonk. Combinatory reduction systems: introduction and survey. *Theoretical Computer Science*, 121:279–308, 1993.

[31] N. P. Mendler. *Inductive Definition in Type Theory.* PhD thesis, Cornell University, United States, 1987.

[32] N. Nelson. *Type inference and reconstruction for first order dependent types.* PhD thesis, Oregon Graduate Institute of Science and Technology, United States, 1995.

[33] F. Pfenning. Refinement types for logical frameworks. In *Proceedings of the International Workshop on Types for Proofs and Programs*, 1993, `http://www.lfcs.informatics.ed.ac.uk/research/types-bra/proc/`.

[34] F. Pfenning and H. Xi. Eliminating array bound checking through dependent types. In *Proceedings of the ACM Conference on Programming Language Design and Implementation*, SIGPLAN Notices 33(5), 1998.

[35] M. Rusinowitch. On termination of the direct sum of term-rewriting systems. *Information Processing Letters*, 26(2):65–70, 1987.

[36] M. Stefanova. *Properties of Typing Systems.* PhD thesis, Katholiecke Universiteit Nijmegen, The Netherlands, 1998.

[37] M.-O. Stehr. *Programming, Specification, and Interactive Theorem Proving - Towards a Unified Language based on Equational Logic, Rewriting Logic, and Type Theory.* PhD thesis, University of Hamburg, Germany, 2002.

[38] Y. Toyama. Counterexamples to termination for the direct sum of term rewriting systems. *Information Processing Letters*, 25(3):141–143, 1987.

[39] C. Walther. Argument-bounded algorithms as a basis for automated termination proofs. In *Proceedings of the 9th International Conference on Automated Deduction*, Lecture Notes in Computer Science 310, 1988.

[40] D. Walukiewicz-Chrząszcz. Termination of rewriting in the Calculus of Constructions. *Journal of Functional Programming*, 13(2):339–414, 2003.

[41] H. Xi. Dependent types for program termination verification. In *Proceedings of the 16th IEEE Symposium on Logic in Computer Science*, 2001.

[42] H. Xi. Dependent types for program termination verification. *Journal of Higher-Order and Symbolic Computation*, 15(1):91–131, 2002.

[43] C. Zenger. Indexed types. *Theoretical Computer Science*, 187(1–2):147–165, 1997.

9 Elimination of transitivity

In this section, we prove Theorem 3 by following Chen's technique [15].

Lemma 41 $\leq$ is equivalent to the relation $\leq'$ where (symb) is replaced by:

$$\text{(symb')} \quad \frac{C^b\vec{t} \leq T}{C^a\vec{t} \leq T} \quad (a \leq_{\mathcal{A}} b)$$

Proof. $\leq\subseteq\leq'$: Assume that $a \leq_{\mathcal{A}} b$. By (refl), $C^b\vec{t} \leq' C^b\vec{t}$. Hence, by (symb'), $C^a\vec{t} \leq' C^b\vec{t}$. $\leq'\subseteq\leq$: Assume that $C^a\vec{t} \leq' T$ since $C^b\vec{t} \leq' T$ and $a \leq_{\mathcal{A}} b$. By induction hypothesis, $C^b\vec{t} \leq T$. By (symb), $C^a\vec{t} \leq C^b\vec{t}$. Therefore, by (trans), $C^a\vec{t} \leq T$. ■

Note that the following two subtyping rules are clearly admissible:

$$\text{(left)} \quad \frac{T \downarrow T' \quad T' \leq U}{T \leq U}$$

$$\text{(right)} \quad \frac{T \leq U' \quad U' \downarrow U}{T \leq U}$$

For representing the subtyping deductions, we introduce the following term algebra:

$$d ::= \bot \mid I \mid Sd \mid Cd \mid Ld \mid Rd \mid Pdd \mid Tdd$$

where $\bot$ stands for some impossible case, I for (refl), S for (symb'), C for (conv), L for (left), R for (right), P for (prod), and T for (trans).

We now prove that the transformation rules of Figure 10 are valid, that is, a deduction matching a left hand-side can be replaced by the corresponding right hand-side.

(a) $Cx \to R(Lx)$

$$\frac{T \downarrow T' \quad T' \leq U' \quad U' \downarrow U}{T \leq U}\, C$$

can be transformed into:

$$\dfrac{\dfrac{T \downarrow T' \quad T' \leq U'}{T \leq U'}\, L \qquad U' \downarrow U}{T \leq U}\, R$$

(b) $R(Rx) \to Rx$

$$\dfrac{\dfrac{T \leq U' \quad U' \downarrow U}{T \leq U}\, R \qquad U \downarrow U''}{T \leq U''}\, R$$

can be transformed into:

$$\dfrac{T \leq U' \quad U' \downarrow U''}{T \leq U''}\, R$$

by confluence of $\to$.

(c) $L(Lx) \to Lx$

Like (b).

(d) $L(Rx) \to R(Lx)$

$$\dfrac{T \downarrow T' \qquad \dfrac{T' \leq U' \quad U' \downarrow U}{T' \leq U}\, R}{T \leq U}\, L$$

can be transformed into:

$$\dfrac{\dfrac{T \downarrow T' \quad T' \leq U'}{T \leq U'}\, L \qquad U' \downarrow U}{T \leq U}\, R$$

Note that the inverse transformation $R(Lx) \to L(Rx)$ is valid too.

(e) $TIx \to x$

$$\dfrac{\dfrac{}{T \leq T}\, I \qquad T \leq U}{T \leq U}\, T$$

can be transformed into:

$$T \leq U$$

(f) $T(Sx)y \to S(Txy)$

$$\dfrac{\dfrac{C^b \vec{t} \leq T}{C^a \vec{t} \leq T}\, S \qquad T \leq U}{C^a \vec{t} \leq U}\, T$$

can be transformed into:

$$\dfrac{\dfrac{C^b \vec{t} \leq T \quad T \leq U}{C^b \vec{t} \leq U}\, T}{C^a \vec{t} \leq U}\, S$$

(g) $T(Lx)y \to L(Txy)$

$$\dfrac{\dfrac{T \downarrow T' \quad T' \leq U}{T \leq U}\, L \qquad U \leq V}{T \leq V}\, T$$

can be transformed into:

$$\dfrac{T \downarrow T' \qquad \dfrac{T' \leq U \quad U \leq V}{T' \leq V}\, T}{T \leq V}\, L$$

(h) $T(RI)x \to Lx$

$$\dfrac{\dfrac{\dfrac{}{T \leq T}\, I \qquad T \downarrow T'}{T \leq T'}\, R \qquad T' \leq U}{T \leq U}\, T$$

can be transformed into:

$$\dfrac{T \downarrow T' \quad T' \leq U}{T \leq U}\, L$$

(i) $T(R(Sx))y \to S(T(Rx)y)$

$$\dfrac{\dfrac{\dfrac{C^b\vec{t} \leq T}{C^a\vec{t} \leq T}\, S \qquad T \downarrow T'}{C^a\vec{t} \leq T'}\, R \qquad T' \leq U}{C^a\vec{t} \leq U}\, T$$

can be transformed into:

$$\dfrac{\dfrac{C^b\vec{t} \leq T \quad T \downarrow T'}{C^b\vec{t} \leq T'}\, R \qquad T' \leq U}{\dfrac{C^b\vec{t} \leq U}{C^a\vec{t} \leq U}\, S}\, T$$

(j) $T(R(Lx))y \to L(T(Rx)y)$

By combination of (g) and the inverse of (d).

(k') $TxI \to x$

Like (e).

(l) $T(R(Pxy))(Sz) \to \bot$

$$\dfrac{\dfrac{\dfrac{U' \leq U \quad V \leq V'}{(x:U)V \leq (x:U')V'}\, P \qquad (x:U')V' \downarrow C^a\vec{t}}{(x:U)V \leq C^a\vec{t}}\, R \qquad \dfrac{C^b\vec{t} \leq T}{C^a\vec{t} \leq T}\, S}{(x:U)V \leq T}\, T$$

is not possible since $(x:U')V'$ and $C^a\vec{t}$ have no common reduct since C is constant.

(n') $Tx(Ry) \to R(Txy)$

$$\dfrac{T \le U \qquad \dfrac{U \le V' \quad V' \downarrow V}{U \le V}\,R}{T \le V}\,T$$

can be transformed into:

$$\dfrac{\dfrac{T \le U \quad U \le V'}{T \le V'}\,T \qquad V' \downarrow V}{T \le V}\,R$$

(m') $T(Rx)(Ly) \to Tx(Ly)$

$$\dfrac{\dfrac{T \le U \quad U \downarrow U'}{T \le U'}\,R \qquad \dfrac{U' \downarrow U'' \quad U'' \le V}{U' \le V}\,L}{T \le V}\,T$$

can be transformed into:

$$\dfrac{T \le U \qquad \dfrac{U \downarrow U'' \quad U'' \le V}{U \le V}\,L}{T \le V}\,T$$

by confluence of $\to$.

(p) $T(R(Pxy))(Pzt) \to P(Tz(Lx))(Ty(Lt))$

$$\dfrac{\dfrac{\dfrac{U_2 \le U_1 \quad V_1 \le V_2}{(x:U_1)V_1 \le (x:U_2)V_2}\,P \qquad (x:U_2)V_2 \downarrow (x:U_3)V_3}{(x:U_1)V_1 \le (x:U_3)V_3}\,R \qquad \dfrac{U_4 \le U_3 \quad V_3 \le V_4}{(x:U_3)V_3 \le (x:U_4)V_4}\,P}{(x:U_1)V_1 \le (x:U_4)V_4}\,T$$

can be transformed into:

$$\dfrac{\dfrac{U_4 \le U_3 \qquad \dfrac{U_3 \downarrow U_2 \quad U_2 \le U_1}{U_3 \le U_1}\,L}{U_4 \le U_1}\,T \qquad \dfrac{V_1 \le V_2 \qquad \dfrac{V_2 \downarrow V_3 \quad V_3 \le V_4}{V_2 \le V_4}\,L}{V_1 \le V_4}\,T}{(x:U_1)V_1 \le (x:U_4)V_4}\,P$$

(r) $T(Pxy)(Sz) \to \bot$

Like (l).

(s') $Tx(LI) \to Rx$

Like (h).

(t) $T(Pxy)(L(Sz)) \to \bot$

Like (l).

(u) $T(Pxy)(L(Pzt)) \to P(Tz(Lx))(Ty(Lt))$

Like (p).

(w) $T(Pxy)(Pzt) \to P(Tzx)(Tyt)$

Like (p).

The above rules form a terminating rewrite system. For L and R, the recursive calls are strictly smaller (take $L < R$). For Tuv, the measure $(|u|+|v|, |v|)$, where $|u|$ is the size of u, strictly decreases lexicographically. Now, it is easy to see that T occurs in no normal form of Tuv if u and v are closed terms (T is completely defined). We proceed by induction on the measure. The only undefined cases for T are $T(R(Pxy))(Tzt)$, $T(Pxy)(L(Tzt))$, $T(Pxy)(Tzt)$ and $T(Txy)z$. By induction hypothesis, T occurs in no normal form of Tzt or Txy. Therefore, we fall in the defined cases and we can conclude by induction hypothesis.

10 Expansion elimination

In this section, we prove Theorem 5 by following Chen's technique [15]. We introduce the following term algebra for representing the subtyping deductions:

$$d ::= I \mid S \mid Ed \mid Rd \mid Pdd$$

where $\perp$ stands for some impossible case, I for (refl), S for (symb), C for (conv), E for (exp), R for (red), and P for (prod).

We now prove that the following transformation rules are valid, that is, a deduction matching a left hand-side can be replaced by the corresponding right hand-side.

$$\begin{array}{lrcl}
(a) & E(Rx) & \to & R(Ex) \\
(b) & E(Pxy) & \to & P(Ex)(Ey) \\
(c) & EI & \to & RI \\
(d) & ES & \to & RS \\
(e) & E(Ex) & \to & Ex
\end{array}$$

(a) $E(Rx) \to R(Ex)$

Assume that we have the following deduction:

$$\dfrac{\dfrac{T' \to^* T'' \leq U'' \ {}^*\!\leftarrow U'}{T \ {}^*\!\leftarrow T' \leq U' \to^* U}\, R}{T \leq U}\, E$$

By confluence, there exist T''' and U''' such that $T \to^* T''' \ {}^*\!\leftarrow T''$ and $U \to^* U''' \ {}^*\!\leftarrow U''$. So, the deduction can be transformed into:

$$\dfrac{\dfrac{T''' \ {}^*\!\leftarrow T'' \leq U'' \to^* U'''}{T \to^* T''' \leq U''' \ {}^*\!\leftarrow U}\, E}{T \leq U}\, R$$

(b) $E(Pxy) \to P(Ex)(Ey)$

Assume that we have the following deduction:

$$\dfrac{\dfrac{C \leq A \quad B \leq D}{T \ {}^*\!\leftarrow (x:A)B \leq (x:C)D \to^* U}\, P}{T \leq U}\, E$$

Then, $T = (x : A')B'$ with $A \to^* A'$ and $B \to^* B'$, and $U = (x : C')D'$ with $C \to^* C'$ and $D \to^* D'$. So, the deduction can be transformed into:

$$\dfrac{\dfrac{C' \ {}^*\!\leftarrow C \leq A \to^* A'}{C' \leq A'}\,E \qquad \dfrac{B' \ {}^*\!\leftarrow B \leq D \to^* D'}{B' \leq D'}\,E}{T \leq U}\,P$$

(c) $EI \to RI$

By confluence, as in (a) but with $T' = T'' = U'' = U'$.

(d) $ES \to RS$

Assume that we have the following deduction:

$$\dfrac{\dfrac{a \leq_{\mathcal{A}} b}{T \ {}^*\!\leftarrow C^a\vec{t} \leq C^b\vec{t} \to^* U}\,S}{T \leq U}\,E$$

Then, $T = C^a\vec{u}$ with $\vec{t} \to^* \vec{u}$ and $U = C^b\vec{v}$ with $\vec{t} \to^* \vec{v}$. By confluence, there exists $\vec{w}$ such that $\vec{u} \to^* \vec{w} \ {}^*\!\leftarrow \vec{v}$. So, the deduction can be transformed into:

$$\dfrac{\dfrac{a \leq_{\mathcal{A}} b}{T \to^* C^a\vec{w} \leq C^b\vec{w} \ {}^*\!\leftarrow U}\,S}{T \leq U}\,R$$

(e) $E(Ex) \to Ex$

Immediate.

Now, the rewrite system defined by these transformation rules is clearly terminating and confluent (there is no critical pair). Since it defines E completely, no normal form of a closed term may contain E.

Figure 10: Transformation rules for eliminating transitivity

$$
\begin{array}{lrcl}
(a) & Cx & \to & R(Lx) \\
(b) & R(Rx) & \to & Rx \\
(c) & L(Lx) & \to & Lx \\
(d) & L(Rx) & \to & R(Lx) \\
(e) & TIx & \to & x \\
(f) & T(Sx)y & \to & S(Txy) \\
(g) & T(Lx)y & \to & L(Txy) \\
(h) & T(RI)x & \to & Lx \\
(i) & T(R(Sx))y & \to & S(T(Rx)y) \\
(j) & T(R(Lx))y & \to & L(T(Rx)y) \\
(k) & T(R(Pxy))I & \to & R(Pxy) \\
(l) & T(R(Pxy))(Sz) & \to & \bot \\
(m) & T(R(Pxy))(Lz) & \to & T(Pxy)(Lz) \\
(n) & T(R(Pxy))(Rz) & \to & R(T(R(Pxy))z) \\
(p) & T(R(Pxy))(Pzt) & \to & P(Tz(Lx))(Ty(Lt)) \\
(q) & T(Pxy)I & \to & Pxy \\
(r) & T(Pxy)(Sz) & \to & \bot \\
(s) & T(Pxy)(LI) & \to & R(Pxy) \\
(t) & T(Pxy)(L(Sz)) & \to & \bot \\
(u) & T(Pxy)(L(Pzt)) & \to & P(Tz(Lx))(Ty(Lt)) \\
(v) & T(Pxy)(Rz) & \to & R(T(Pxy)z) \\
(w) & T(Pxy)(Pzt) & \to & P(Tzx)(Tyt) \\
(1) & S\bot & \to & \bot \\
(2) & L\bot & \to & \bot \\
(3) & R\bot & \to & \bot \\
(4) & P\bot x & \to & \bot \\
(5) & Px\bot & \to & \bot \\
(6) & T\bot x & \to & \bot \\
(7) & Tx\bot & \to & \bot
\end{array}
$$

Some of these rules are particular instances of the following more general transformations:

$$
\begin{array}{lrcl}
(k')(q') & TxI & \to & x \\
(n')(v') & Tx(Ry) & \to & R(Txy) \\
(m') & T(Rx)(Ly) & \to & Tx(Ly) \\
(s') & Tx(LI) & \to & Rx
\end{array}
$$